Lior Rokach
Oded Maimon
Erez Shmueli *Editors*

Machine Learning for Data Science Handbook

Data Mining and Knowledge Discovery Handbook

Third Edition

Springer

Machine Learning for Data Science Handbook

Lior Rokach • Oded Maimon • Erez Shmueli
Editors

Machine Learning for Data Science Handbook

Data Mining and Knowledge Discovery Handbook

Third Edition

Springer

Editors
Lior Rokach
Department of Software and Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva, Israel

Oded Maimon
Department of Industrial Engineering
Tel Aviv University
Ramat Aviv, Israel

Erez Shmueli
Department of Industrial Engineering
Tel Aviv University
Tel Aviv, Israel

ISBN 978-3-031-24627-2 ISBN 978-3-031-24628-9 (eBook)
https://doi.org/10.1007/978-3-031-24628-9

1st edition: Springer-Verlag US 2005
2nd edition: Springer Science+Business Media, LLC 2010
3rd edition: © Springer Nature Switzerland AG 2023

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Data Science and Knowledge Discovery Using Machine Learning Methods

Oded Maimon, Lior Rokach, and Erez Shmueli

1 Introduction

Since the dawn of the big data age, accumulating and storing data has become more accessible and inexpensive. Data science is an emerging interdisciplinary field that combines methods, processes, technologies, and know-how from various fields, particularly statistics, data mining, machine learning, big data, and business intelligence, to address data-driven problems. Data science usually involves discovering knowledge and actionable insights from data and then applying them to solve problems in various domains. /textit[Knowledge Discovery in Databases] (KDD) is another closely related term referring to automatic, exploratory analysis and modeling of large data repositories. The goal of KDD is to identify novel, useful, valid, and understandable patterns in large and complex data sets. Data mining (DM) is an integral part of the KDD process, involving algorithms that explore data, develop models, and uncover previously unknown patterns in order to make predictions and understand phenomena that are found in the data sets.

Today's accessibility and abundance of data make knowledge discovery and data science matters of considerable importance and necessity. Given the field's recent growth, it is not surprising that researchers and practitioners now have a wide range of methods and tools at their disposal.

While statistics is fundamental for data science, methods originated from artificial intelligence, particularly machine learning, are also playing a significant role.

O. Maimon · E. Shmueli
Department of Industrial Engineering, Tel-Aviv University, Tel Aviv-Yafo, Israel
e-mail: maimon@tauex.tau.ac.il; shmueli@tauex.tau.ac.il

L. Rokach (✉)
Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Be'er Sheva, Israel
e-mail: liorrk@bgu.ac.il; liorrk@post.bgu.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_1

Artificial intelligence (AI) is a scientific discipline that aims to create intelligent machines. Machine learning is a popular AI subfield that seeks to improve the performance of computer programs at a given task through experience rather than being explicitly programmed. Most machine learning techniques are based on inductive learning , where a model is constructed explicitly or implicitly by generalizing from a sufficient amount of training data. Machine learning methods are becoming increasingly popular in data science and data mining.

This handbook is intended to organize all significant methods developed in the field into a coherent and unified catalog. It presents approaches and techniques for performance evaluation and shows the different techniques with examples and software tools. The target audiences of the handbook include students, data scientists, ML engineers, researchers, and practitioners.

This introductory chapter aims to explain the KDD process and position machine learning within this process. Research and development challenges for the next generation of data science are also defined. The rationale, reasoning, and organization of the handbook are presented in this chapter for helping the reader to navigate the extremely rich and detailed content provided in this handbook.

In this chapter, there are six sections: 1. The KDD process; 2. Taxonomy of data mining methods; 3. Data mining within the complete decision support system; 4. Data science and KDD research opportunities and challenges; 5. Recent trends in data science; 6. The organization of the handbook

The unique recent aspects of data availability that promote the rapid development of data science are the electronic readiness of data (though of different types and reliability). In particular, the Internet and Intranet's fast growth promotes data accessibility (as formatted or unformatted, voice or video, etc.). Methods developed before the Internet revolution considered smaller amounts of data with less variability in data types and reliability. Since the information age, the accumulation of data has become more accessible and less costly. It has been estimated that stored information doubles every twenty months. Unfortunately, as the amount of electronically stored information increases, the ability to understand and make use of it does not keep pace with its growth. Data mining is a term coined to describe the process of sifting through large databases for interesting patterns and relationships. Today's studies aim at evidence-based modeling and analysis, as is the leading practice in healthcare, finance, cyber-security, and many other fields. Data availability increases exponentially, while the human processing level is almost constant. Thus, the potential gap rises exponentially. This gap is the opportunity for the data science field, which has become increasingly important and necessary.

2 The KDD Process

The knowledge discovery process (Fig. 1) is iterative and interactive, consisting of nine steps. Note that the process is iterative at each stage, meaning that moving back to adjust previous steps may be required. The process has many "artistic" aspects in the sense that one cannot present one formula or make a complete taxonomy

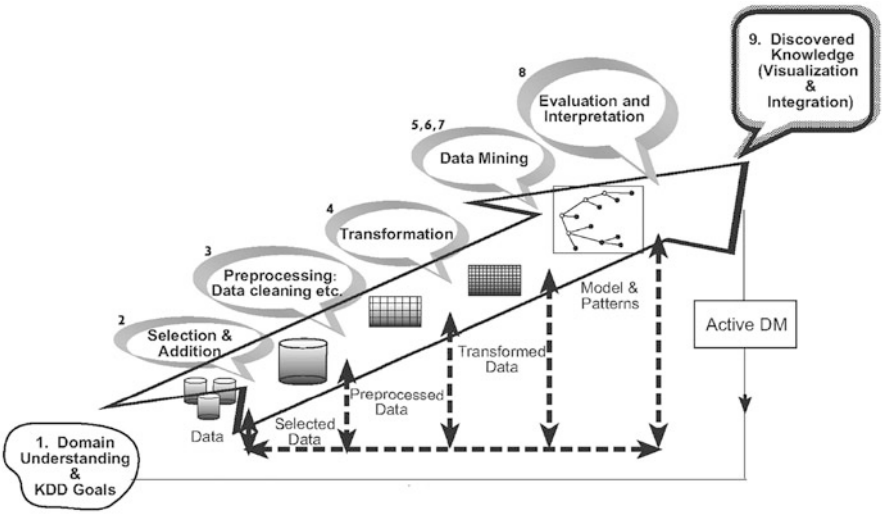


Fig. 1 The process of knowledge discovery in databases

for the right choices for each step and application type. Thus, it is required to deeply understand the process and the different needs and possibilities in each step. Taxonomy for the data science methods is helping in this process. It is presented in the next section.

The process starts with determining the KDD goals and "ends" with the implementation of the discovered knowledge. As a result, changes would have to be made in the application domain (such as offering different features to mobile phone users to reduce churning). Such modification closes the loop, the effects are then measured on the new data repositories, and the KDD process is relaunched. Following is a brief description of the nine-step KDD process, starting with a managerial step:

1. Developing an understanding of the application domain The people in charge of a KDD project need to understand and define the end-user goals and the environment in which the knowledge discovery process will take place (including relevant prior knowledge). As the KDD process proceeds, there may be a revision and tuning of this step. Having understood the KDD goals, the preprocessing of the data starts, as defined in the next three steps (note that some of the methods here are similar to data mining algorithms but are used in the preprocessing context).
2. Selecting and creating a data set on which we will work. Having defined the goals, the data used for the knowledge discovery should be determined. This includes finding out what data are available, obtaining additional necessary data, and then integrating all the data for the knowledge discovery into one data set, including the attributes considered for the process. This process is essential because data mining learns and discovers from the available data. This

is the evidence base for constructing the models. If some crucial attributes are missing, then the entire study may fail. From the success of the process, it is good to consider as many as possible attributes at this stage. On the other hand, collecting, organizing, and operating complex data repositories are expensive, and there is a tradeoff with the opportunity to understand the phenomena best. This tradeoff represents an aspect where the interactive and iterative part of the KDD is taking place. It starts with the best available data set and later expands and observes the effect in terms of knowledge discovery and modeling.

3. Preprocessing and cleansing. In this stage, data reliability is enhanced. It includes data clearing, such as handling missing values and removing noise or outliers. Several methods are explained in the handbook, from doing nothing to becoming the major part (in terms of time consumed) of a KDD process in specific projects. It may involve complex statistical methods or specific data mining algorithms in this context. For example, suppose one suspects that a particular attribute is not reliable enough or has too much missing data. In that case, this attribute could become the goal of a data mining supervised algorithm. A prediction model for this attribute will be developed, and then missing data can be predicted. The extent to which one pays attention to this level depends on many factors. Studying these aspects is essential and often reveals insights by itself regarding enterprise information systems.
4. Data transformation. In this stage, the generation of better data for the data mining is prepared and developed. Methods here include dimension reduction (such as feature selection and extraction, and record sampling) and attribute transformation (such as discretization of numerical attributes and functional transformation). This step is often crucial for the success of the entire KDD project, but it is usually very project-specific. For example, in medical examinations, the quotient of attributes may often be the most important factor and not each one by itself. In marketing, we may need to consider effects beyond our control as well as efforts and temporal issues (such as studying the impact of advertising accumulation). However, even if we do not use the right transformation initially, we may obtain a surprising effect that hints to us about the transformation needed (in the next iteration). Thus the KDD process reflects upon itself and leads to an understanding of the necessary transformation (like a concise knowledge of an expert in a particular field regarding key leading indicators). Having completed the above four steps, the following four steps are related to the data mining part, where the focus is on the algorithmic aspects employed for each project.
5. Choosing the appropriate data science task. We are now ready to decide on which type of data science task to perform, for example, classification, regression, or clustering. This mostly depends on the KDD goals and also on the previous steps. There are two major tasks in data science: prediction and description. Prediction is often referred to as supervised machine learning and statistical regression analysis, while descriptive includes exploratory data analysis, statistical hypothesis testing, unsupervised machine learning, and visualization aspects. Many data science techniques are based on inductive

learning, where a model is constructed explicitly or implicitly by generalizing from a sufficient number of training examples. The underlying assumption of the inductive approach is that the trained model applies to future cases. The strategy also considers the level of meta-learning for the particular set of available data.

6. Choosing the algorithm. Having the strategy, we now decide on the tactics. This stage includes selecting the specific method for searching patterns (including multiple inducers). For example, considering precision versus understandability, the former is better with neural networks, while the latter is better with decision trees.
7. Running the algorithm. Finally, the implementation of the algorithm is reached. In this step, we might need to run the algorithm several times until a satisfying result is obtained, for instance, by turning the algorithm's control parameters, such as the minimum number of instances, in a single leaf of a decision tree.
8. Evaluation. In this stage, we evaluate the predictive performance of the trained model and interpret the mined patterns (rules, reliability, etc.) with respect to the goals defined in the first step. Here we consider the preprocessing steps concerning their effect on the results (for example, adding features in Step 4 and repeating from there). This step focuses on the comprehensibility and usefulness of the induced model. In this step, the discovered knowledge is also documented for further usage. The last step is the usage and overall feedback on the patterns and discovery results obtained by the models.
9. Using the discovered knowledge. We are now ready to incorporate the knowledge into another system for further action. The knowledge becomes active because we may make changes to the system and measure the effects. The success of this step determines the effectiveness of the entire KDD process. There are many challenges in this step, such as losing the "laboratory conditions" under which we have operated. For instance, the knowledge was discovered from a certain static snapshot (usually sample) of the data, but now the data become dynamic. Data structures may change (certain attributes become unavailable), and the data domain may be modified (such as an attribute may have a value that was not assumed before).

3 Taxonomy of Data Science Methods

Depending on the main purpose, many methods are used in practice for data science. Taxonomy is called for to help understand the variety of methods, their interrelation, and grouping. It is useful to distinguish between two main types of data science: verification-oriented (the system verifies the user's hypothesis) and discovery-oriented (the system finds new rules and patterns autonomously). Figure 2 presents this taxonomy.

Discovery methods are those that automatically identify patterns in the data. The discovery method branch consists of prediction methods versus description

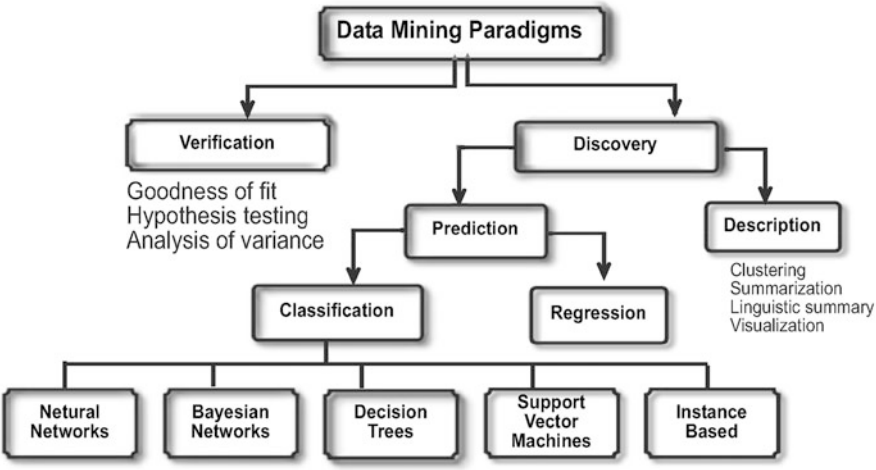


Fig. 2 Data science taxonomy

methods. Descriptive methods are oriented to data interpretation, which focuses on understanding (by visualization, for example) the way the underlying data relate to its parts. Prediction-oriented methods aim to build automatically a behavioral model that obtains new and unseen samples and can predict the values of one or more variables related to the sample. It also develops patterns, which form the discovered knowledge in a way that is understandable and easy to operate upon. Some prediction-oriented methods can also help provide an understanding of the data.

Recall that most of the discovery-oriented techniques (quantitative in particular) are based on inductive learning, where a model is constructed, explicitly or implicitly, by generalizing from a sufficient number of training examples. The underlying assumption of the inductive approach is that the trained model applies to future unseen examples.

Verification methods, on the other hand, deal with the evaluation of a hypothesis proposed by an external source (like an expert, etc.). These methods include the most common methods of traditional statistics, such as the goodness-of-fit test, tests of hypotheses (e.g., t-test of means), and analysis of variance (ANOVA). These methods are less associated with data mining than their discovery-oriented counterparts because most data mining problems are concerned with discovering a hypothesis (out of a very large set of hypotheses), rather than testing a known one. Much of the focus of traditional statistical methods are on model estimation as opposed to one of the main objectives of data science: model identification and construction, which is evidence -based (though overlap occurs). There can be a security flaw, for example, an unauthorized user accessing an application or a specific feature that s/he is not supposed to access, or a malicious user attempting to do something in order to break the system. These kinds of flaws can not be

acceptable. Unsupervised learning refers to modeling the distribution of instances in typical, high-dimensional input space.

Unsupervised learning refers mostly to techniques that group instances without a prespecified, dependent attribute. Thus the term "unsupervised learning" covers only a portion of the description methods presented in Fig. 2. For instance, it covers clustering methods but not visualization methods. Supervised methods are methods that attempt to discover the relationship between input attributes (sometimes called independent variables) and a target attribute (sometimes referred to as a dependent variable). The association discovered is represented in a structure referred to as a model. Usually, models describe and explain phenomena, which are hidden in the data set and can be used for predicting the value of the target attribute, knowing the values of the input attributes. The supervised methods can be implemented on a variety of domains,, such as marketing, finance, and manufacturing. It is useful to distinguish between two main supervised models: classification models and regression models. The latter map the input space into a real-valued domain. For instance, a regressor can predict the demand for a certain product given its characteristics. On the other hand, classifiers map the input space into predefined classes. For example, classifiers can be used to classify mortgage consumers as good (fully payback the mortgage on time) and bad (delayed payback), or as many target classes as needed. There are many alternatives to represent classifiers. Typical examples include support vector machines, decision trees, probabilistic summaries, or algebraic functions.

4 Data Science Within the Complete Decision Support System

Data science methods are becoming part of integrated information technology (IT) software packages. Figure 3 illustrates the three tiers of the decision support aspect of IT. Starting from the data sources (such as operational databases, semi- and non-structured data and reports, Internet sites, etc.), the first tier is the data warehouse, followed by OLAP (On-Line Analytical Processing) servers and concluding with analysis tools, where data science tools are the most advanced.

The main advantage of the integrated approach is that the preprocessing steps are much easier and more convenient. Since this part is often the major burden for the KDD process (and can consume most of the KDD project time), this industry trend is very important for expanding the use and utilization of data mining. However, the risk of the integrated IT approach comes from the fact that DM techniques are much more complex and intricate than OLAP, for example, so the users need to be trained appropriately.

This handbook shows the variety of strategies, techniques, and evaluation measurements. We can naively distinguish among three levels of analysis. The simplest one is achieved by report generators (for example, presenting all claims that

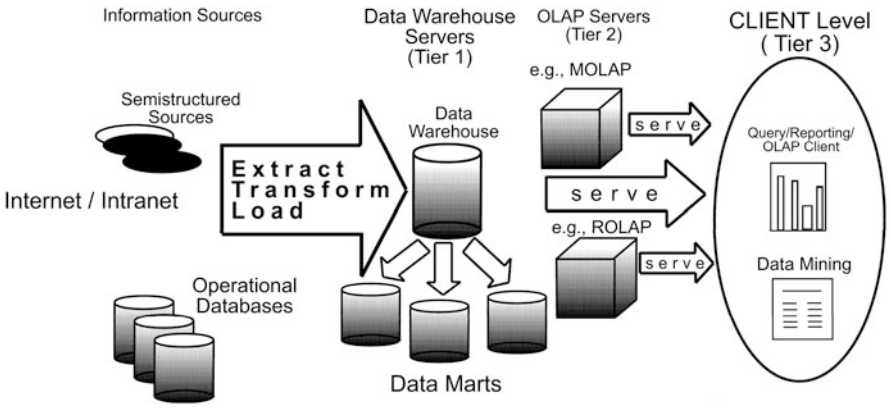


Fig. 3 The IT decision support tiers

occurred because of a specific cause last year, such as car theft). We then proceed to OLAP multi-level analysis (for example, presenting the ten towns where there was the highest increase of vehicle theft in the last month compared to the month before). Finally, a complex analysis is carried out in discovering the patterns that predict car thefts in these cities and what might occur if anti -theft devices were installed. The latter is based on mathematical modeling of the phenomena, where the first two levels are ways of data aggregation and fast manipulation. This handbook mainly focuses on the third level of analysis.

5 KDD and Data Science Research Opportunities and Challenges

An empirical comparison of the performance of different approaches and their variants in a wide range of application domains has shown that each performs best in some, but not all, domains. This phenomenon is known as the selective superiority problem, which means, in our case, that no induction algorithm can be the best in all possible domains. The reason is that each algorithm contains an explicit or implicit bias that leads it to prefer certain generalizations over others, and it will be successful only as long as this bias matches the characteristics of the application domain. Results have demonstrated the existence and correctness of this "no free lunch theorem." If one inducer is better than another in some domains, then there are necessarily other domains in which this relationship is reversed. This implies in KDD that a certain approach can yield more knowledge from the same data for a given problem than other approaches.

In many application domains, the generalization error (on the overall domain, not just the one spanned in the given data set) of even the best methods is far above

the training set, and the question of whether it can be improved, and if so how, is an open and important one. Part of the answer to this question is to determine the minimum error achievable by any classifier in the application domain (known as the optimal Bayes error). If existing classifiers do not reach this level, new approaches are needed. Although this problem has received considerable attention, no generally reliable method has so far been demonstrated. This is one of the challenges of data science research—not only to solve it, but even to quantify and understand it better. Heuristic methods can then be compared absolutely and not just against each other.

A subset of this generalized study is the question of which inducer to use for a given problem. To be even more specific, the performance measure needs to be defined appropriately for each problem. Though there are some commonly accepted measures, it is not enough. For example, if the analyst is looking for accuracy only, one solution is to try each one in turn, and by estimating the generalization error, choose the one that appears to perform best. Another approach, known as multi-strategy learning, attempts to combine two or more different paradigms in a single algorithm. The dilemma of which method to choose becomes even greater if other factors, such as comprehensibility, are taken into consideration. For instance, neural networks may outperform decision trees in accuracy for a specific domain; however, from the comprehensibility aspect, decision trees are considered superior. In other words, even if the researcher knows that neural network is more accurate, the dilemma of what methods to use still exists (or maybe to combine techniques for their different strengths).

Induction is one of the central problems in many disciplines, such as machine learning, pattern recognition, and statistics. However , the feature that distinguishes data science from statistics is its scalability to very large sets of varied types of input data. Scalability means working in an environment of a high number of records, high dimensionality, and an increased number of classes or heterogeneousness. Nevertheless, trying to discover knowledge in real life and large databases introduces time and memory problems. As large databases have become the norm in many fields (including astronomy, molecular biology, finance, marketing, health care, and many others), the use of data science to discover patterns in them has become potentially very beneficial for the enterprise. Many companies are staking a large part of their future on these "data science" applications and turn to the research community for solutions to the fundamental problems they encounter. While a very large amount of available data used to be the dream of any data analyst, nowadays the synonym for "very large" has become "terabyte" or "petabyte," a barely imaginable volume of information.

Information-intensive organizations (like telecom companies and financial institutions) are expected to accumulate several terabytes of raw data every one to two years. The high dimensionality of the input (that is, the number of attributes) increases the size of the search space in an exponential manner (known as the "Curse of Dimensionality") and thus increases the chance that the inducer will find spurious classifiers that in general are not valid. There are several approaches for dealing with a high number of records including: sampling methods, aggregation, massively parallel processing, and efficient storage methods.

6 KDD and DM Trends

This handbook covers the current state-of-the-art status of data science. The field is still evolving in a sense that some new methods are still being developed. The art expands, but so does the understanding and the automation of the nine steps and their interrelation. For this to happen, we need better characterization of the KDD problem spectrum and definition. The terms KDD and data science are not well-defined in terms of what methods they contain, what types of problems are best solved by these methods, and what results to expect. How are KDD\data science compared to statistics, machine learning, etc.? If subset or superset of the above fields? Or an extension\adaptation of them? In addition to the methods—which are the most promising fields of application and what is the vision KDD\data science brings to these fields? Certainly, we already see the great results and achievements of data science, but we cannot estimate their results concerning the potential of this field. All these basic analyses have to be studied, and we see several trends for future research and implementation, including:

- Active data science—closing the loop, as in control theory, where changes to the system are made according to the KDD results, and the whole cycle starts again. Stability and controllability, which will be significantly different in these systems, need to be well-defined.
- Full taxonomy—for all the nine steps of the KDD process. We have shown a taxonomy for the primary step, but taxonomy is needed for each nine steps. Such a taxonomy will contain methods appropriate for each step (even the first one) and for the whole process as well.
- Benefit analysis—to understand the effect of the potential data science models on the enterprise.
- Problem characteristics—analysis of the problem itself for suitability to the KDD process.
- Mining complex objects of arbitrary type—Expanding data science inference to include mixed data types such as pictures, text, voice, video, audio, etc. Multimodal deep learning algorithms that were developed in the last five years are a step forward in addressing this challenge.
- Temporal aspects—many data mining methods assume that discovered patterns are static. However, in practice, patterns in the database evolve over time. This poses two important challenges. The first challenge is to detect when concept drift occurs. The second challenge is to keep the patterns up-to-date without inducing the patterns from scratch.
- Distributed Data Mining—The ability to seamlessly and effectively employ data science methods on databases located in various sites or the cloud. This problem is especially challenging when the data structures are heterogeneous rather than homogeneous.
- Expanding data science reasoning to include models that address not just data that look similar to the training data, but also out-of-distribution data.

7 The Organization of the Handbook

This handbook is organized into eight parts described in the following sub-sections. Starting from Chap. "Handling Missing Attribute Values" through to the end of part six, the book presents a comprehensive but concise description of different methods used throughout the KDD process. Each part describes the classic techniques as well as the extensions and novel methods developed recently. Along with the algorithmic description of each technique, the reader is provided with an explanation of the circumstances in which this method is applicable and the consequences and the tradeoffs of using the method, including references for further readings. Part seven presents domain-specific applications. The last part discusses human factors and social issues that should be considered while doing data science.

7.1 Data Preparation Methods

The first part deals with data preparation. This covers the preprocessing methods (Steps 3, 4 of the KDD process). Chapter "Handling Missing Attribute Values" presents various methods for handling missing values. These methods are categorized into two main types: sequential and parallel. In sequential methods, missing attribute values are replaced by known values first, as a preprocessing, and then the knowledge is acquired for a data set with all known attribute values. In parallel methods, there is no preprocessing, i.e., knowledge is acquired directly from the original data sets.

Automated data integration, specifically, ETL (Extract-Transform-Load), is one of the most important step in building a data warehouse (DWH). In Chap. "Data Integration Process Automation using Machine Learning: Issues and Solution", the solution approach of the automated ETL process is explained. It also describes how machine learning can be leveraged in the ETL process so that the quality and availability of data not ever have been compromised.

7.2 Supervised Learning

The classic supervised methods are presented in the second part (not including deep learning that will be discussed separately in part 4). Chapter "Rule Induction" discuses rule induction. It begins with a brief discussion of some problems associated with input data. Then, four representative rule induction algorithms are presented: LEM,1, LEM2, MLEM2, and AQ. Finally, some more advanced methods are listed.

One of the first non-parametric supervised learning methods developed was nearest neighbors. Although these methods are considered simple, they remain

competitive in some cases against state-of-the-art techniques. Chapter "Nearest-Neighbor Methods: A Modern Perspective" aims at providing an overview of various modern approaches to learning with nearest neighbors in general metric spaces. Kontorovich and Kpotufe provide the necessary background and then proceed to cover classification and regression. The techniques are described with sufficient detail and literature references to provide practical insights into how various configuration and preprocessing choices, such as the metric, the number of neighbors, data subsampling, and compression, influence learning and computational performance.

Support vector machines (SVMs) are methods for supervised learning, applicable to both classification and regression tasks. For example, an SVM classifier creates a maximum-margin hyperplane in transformed input space and splits the example classes while maximizing the distance to the nearest cleanly split samples. The parameters of the solution hyperplane are derived from a quadratic programming optimization problem. Chapter "Support Vector Machines" focuses on the formulation of SVM models and discusses some key concepts.

A common belief in the machine learning (ML) community is that while supervised learning methods such as decision trees, neural networks (NNs), and SVM methods are the ultimate tools for highly accurate classification, graphical models, and in particular Bayesian networks (BNs), are only appropriate in knowledge representation. Chapter "Empowering Interpretable, Explainable Machine Learning Using Bayesian Network Classifiers" challenges the belief that the unsupervised graphical model is inferior to the supervised classifier and provides evidence to the contrary. Moreover, it manifests how their capability in knowledge representation allows graphical models to promote interpretability and explainability that are not natural to conventional ML classifiers.

Chapter "Soft Decision Trees" introduces the foundation of a new theory for decision- trees-based models. It utilizes the notion of soft numbers, which combines real processes and cognitive ones in the same framework. Moreover, soft numbers offer a new way to deal with uncertainty by incorporating soft numbers into probability theory.

Chapter "Quality Assessment and Evaluation Criteria in Supervised Learning" reviews commonly used predictive performance measures of supervised learning algorithms and discusses their properties. The author presents conceptual tools and provides essential guidelines for quality assessment of fully trained models, particularly classifiers and regression models. Finally, algorithm design considerations are discussed to optimize the desired evaluation criteria.

7.3 Unsupervised Learning Methods

The third part of the handbook is dedicated to unsupervised methods. Chapter "Trajectory Clustering Analysis" discusses the notion of trajectory clustering analysis. Specifically, the authors present a general framework, termed atomic-representation

based subspace clustering (ARSC) for the clustering of trajectory data. ARSC is a subspace clustering framework by first computing the atomic representations of data points and then clustering them using the representations.

Chapter "Clustering High-Dimensional Data" focuses on clustering algorithms that have been adapted or designed explicitly for high-dimensional data. In such cases, many attributes might be just noise such that patterns can be identified only in appropriate combinations of attributes and would be obfuscated by noise otherwise. An overview of the basic strategies and techniques used for these specialized algorithms is provided in this chapter, along with examples of how they can be implemented.

Chapter "Fuzzy C-Means Clustering: Advances and Challenges (Part II)" presents the popular fuzzy C-means (FCM) algorithm followed by a profound discussion of its challenges and recent development, such as improving the optimality condition of cluster fuzziness or efficiently choosing the optimal cluster center. Toward the end of this chapter, a factual analysis has been presented on the applications of FCM in various research domains with their growth.

Data streams are data that continuously arrive over long periods. This precludes the use of conventional methods based on storing the data for later use. Chapter "Clustering in Streams" provides an overview of stream clustering algorithms and their applications to various types of domains. Stream clustering is very common in the online setting because it is often used as a subroutine for other data mining problems. For example, stream clustering is often used to enable methods for classification and anomaly detection.

7.4 Deep Learning

The fourth part discusses the deep learning methods. Deep learning is a type of machine learning that utilizes artificial neural networks with representation learning that consists of multiple layers between the input and output layers. Deep learning (DL) has made a significant impact on data science in the last decade. Chapter "Introduction to Deep Learning" introduces the basic concepts of this field. It includes the fundamental structures used to design deep neural networks and a brief survey of some of its widespread use cases.

Graph embedding aims to represent graphs in a low-dimensional space by capturing various properties of the graphs. Graph embedding is essential for various tasks, including graphs' similarities, time-series trends analysis, and anomaly detection, graph visualization, graph classification, and clustering. Chapter "Graph Embedding" presents three categories of embedding problems: (i) static graphs, (ii) dynamic graphs, and (iii) attributed graphs. The chapter discusses all the state-of-the-art methods along with their applications.

An autoencoder is a specific type of neural network, which is mainly designed to encode the input into a compressed and meaningful representation and then decode it back such that the reconstructed input is similar as possible to the original one.

Chapter "Autoencoders" presents the different types of autoencoders that are mainly used today. It also describes various applications and use cases of autoencoders.

Generative adversarial networks (GANs) aim at generating new high-quality data that have the same properties as the training set. Chapter "Generative Adversarial Networks" provides an introduction to GANs by discussing their principle mechanism and presenting some of their inherent problems during training and evaluation. In this chapter, Gilad Cohen and Raja Giryes focus on these three issues: (1) mode collapse, (2) vanishing gradients, and (3) generation of low-quality images. They then list some architecture-variant and loss-variant GANs that remedy the above challenges. Lastly, they present two utilization examples of GANs for real-world applications: data augmentation and face images generation.

7.5 Methods for Special Data Setting

Having established the foundation, we now proceed with methods developed for specific data settings. Spatial data science is a multidisciplinary field that focuses on the unique characteristics of spatial data. Chapter "Spatial Data Science" discusses spatial data science and describes its life cycle: data acquisition, data storage, data mining, result validation, and domain interpretation. Spatial data science is important for societal applications in public health, public safety, agriculture, environmental science, climate, etc. The challenges of spatial data science are brought about by its interdisciplinary nature and the unique properties of spatial data, such as spatial autocorrelation and spatial heterogeneity.

Multimedia data learning is an emerging, multidisciplinary, and interdisciplinary research area with a broad spectrum of real-world applications. The research in this field focuses on the synergistic applications of knowledge discovery theories and techniques in a multimedia collection. Chapter "Multimedia Data Learning" introduces the fundamental concepts and theories of this area and provides further references.

The World Wide Web (WWW) allows users and organizations to publish information and documents instantly available for all other users of the Web. The data published on the Web continuously increase, providing the users with a vast amount of information on any topic imaginable. However, navigating the Web and identifying the relevant pieces of information in the abundance of data are not trivial. Web mining approaches that analyze the Web data and the Web structure are designed to address these challenges. Chapter "Web Mining" aims at providing a summary of Web mining approaches, including Web content mining, Web structure mining, Web usage mining, and semantic Web mining.

Temporal data refer to data where time and changes over time are crucial for its analysis. They are typically defined by data elements collected repeatedly over time about the object of interest (e.g., a car). Chapter "Mining Temporal Data" presents the various types of temporal data and the various relevant analysis methods: starting with fixed frequency variables, forecasting and time-series methods,

and proceeding with sequential data, sequential patterns mining, and time intervals mining for events having various time duration. Moreover, various deep-learning-based architectures for temporal data are discussed.

7.6 Methods for Special Learning Tasks

The sixth part covers advanced learning tasks. Cloud data mining introduces the concept of performing data mining and analytics of big data in the cloud. Chapter "Cloud Big Data Mining and Analytics: Bringing Greenness and Acceleration in the Cloud" presents four technologies for the acceleration of computing and analysis of data mining tasks in the cloud: graphics processing units (GPU), approximate computing, quantum computing, and neural processing units.

Chapter "Multi-label Ranking: Mining Multi-label and Label Ranking Data" focuses on multi-label ranking tasks, specifically multi-label classification and label ranking classification. The chapter presents the recent developments, focusing on state-of-the-art methods in deep learning multi-label mining, extreme multi-label classification, and label ranking.

Reinforcement learning (RL) is a distinct field of machine learning, in which an agent *can act*, i.e., influence the environment it exists in, and receive rewards (scalar values). It is distinct from supervised learning in that the true output is unknown and from unsupervised learning in that feedback is received from the environment, i.e., the reward. Chapter "Reinforcement Learning for Data Science" aims to introduce the reader to the basic principles, formulations, and algorithms of reinforcement learning (Sect. 2) and give an example of its use in data science (Sect. 3) and explore the state-of-the-art approaches to deep reinforcement learning (Sect. 4).

Adversarial machine learning studies the behavior of machine learning models in the presence of an adversary. Its focus is understanding the susceptibility of machine learning algorithms to specially crafted inputs, referred to as adversarial examples or adversarial perturbations. Chapter "Adversarial Machine Learning" follows the evolution of adversarial machine learning research in recent years, through the lens of the literature. Ziv Katzir and Yuval Elovici begin by reviewing early work on attack and defense methods and move on to studies that show how adversarial attacks can be applied in the real world. Then they list the major outstanding research questions and conclude with research that addresses the domain's key open question: What makes adversarial examples so difficult to defend against?

As indicated in Chap. "Introduction to Deep Learning", deep learning has become a prevalent method for text classification in recent years, due to its ability to improve the accuracy of previous state-of-the-art methods. However, these improvements required hundreds of thousands to millions of labeled training examples, which in many cases can be very time-consuming and expensive to acquire. This problem is especially significant in domain-specific text classification tasks where pre-trained embeddings and models are not optimal. In Chap. "Ensembelled Transferred Embeddings", the ensembled transferred embeddings (ETE) method is

proposed. ETE relies on two ideas: (1) labeling a relatively small sample of the target data set, in a semi-automatic process, and (2) leveraging other data sets from related domains or related tasks that are large scale and labeled, to extract "transferable embeddings."

7.7 Domain-Specific Applications

With all the methods described so far, the next section, the seventh, is concerned with data science applications in various domains such as Healthcare and e-commerce.

Clinical databases collect large volumes of information. Relationships and patterns within these data could provide new medical knowledge. Data mining has as significant objective the discovery of knowledge from large amounts of data and offers many possibilities for identifying different data features less visible or hidden to common analysis techniques. Chapter "Data Mining in Medicine" presents a selection of techniques and illustrates their applicability to medical diagnostic and prognostic problems.

Recommender systems are software tools and techniques providing suggestions for items to be of use to a user. The suggestions provided by a recommender system aim to support their users in various decision-making processes, such as what items to buy, what music to listen to, or what news to read. A personalized recommendation can reduce customers' effort in finding items they are interested in and serve as valuable means for online users to cope with information overload. This is why recommender systems have achieved widespread success in real-life applications. Chapter "Recommender Systems" introduces the fundamentals and advances of recommender systems, including presentation of the widely used techniques, applications, and evaluation methods of recommender systems.

In ubiquitous computing, it is critical to infer human behaviors and activities, which can then be used as information by downstream tasks. Chapter "Activity Recognition" is dedicated to human activity recognition (HAR). First, the chapter briefly introduces the basics of HAR and its applications in ubiquitous computing. Then, it introduces the main procedures of HAR, followed by more detailed components: data preprocessing and feature engineering, model building, and evaluations. Finally, the authors present some grand challenges in HAR that could be improved in the future.

Fake news is a long-lasting problem that has drawn significant attention in recent years. There is a growing need for tools and methods to control the spread of misinformation through online social media. Machine learning methods have been utilized to pinpoint linguistic patterns, influential accounts, or spreading dynamics associated with misinformation. In Chap. "Social Network Analysis for Disinformation Detection", Elyashar et al. present an automated process for training fake news classifiers based on multiple families of features extracted from social media. In addition to the high accuracy of the trained machine learning classifiers, their

results show that online social media users are aware of deceptive content and can often provide reliable feedback to detect fake news.

A closely related challenge to fake news is propaganda dissemination. The Internet has provided new and effective ways of propaganda dissemination such as social network platforms and online forums. Accurate and timely detection of propaganda content remains a highly challenging task. Chapter "Online Propaganda Detection" focuses on automated methods and systems for online propaganda detection. In this chapter, the difficulties of developing such systems and the potential contributions of machine learning are explained.

Chapter "Interpretable Machine Learning for Financial Applications" focuses on machine learning (ML) for financial applications including interpretable relational methods. It presents financial tasks, methodologies, and techniques in this area. In particular, it surveys main methods, including time dependence, data selection, forecast horizon, measures of success, quality of patterns, hypothesis evaluation, and attribute-based and interpretable relational methodologies. The second part of the chapter covers the use of ML in portfolio management, the design of interpretable trading rules, and discovering money-laundering schemes using the machine learning methodology.

Predictive analytics (PA) models are assuming an increasing role in big data for making decisions in many industries such as marketing, banking, insurance, telecommunication, healthcare, and cyber. While regression models were initially developed to explain phenomena, find relationships between variables, and draw conclusions, in prediction models, the main objective is to build models that are general enough to apply for predicting unseen data, even at the expense of giving up some model accuracy. Therefore, models with good explanation power are not necessarily models with good prediction power and vice versa. In Chap. "Predictive Analytics for Targeting Decisions", Zahavi discusses the differences between explanation and prediction models, proposes several principles for building good predictive models, and presents several performance measures for assessing the quality of the prediction results in classification problems using logistic regression. Zahavi concludes by discussing the deployment process of the model results for decision-making and by briefly reviewing the non-parametric decision tree approach for building PA model.

Geoscience phenomena (e.g., earthquakes) are often studied using data-driven models, which are based on various types of data that are monitored and sensed using sophisticated equipment. Chapter "Machine Learning for the Geosciences" provides a review on data-driven problems in geoscience, with a particular focus on the subfield of seismology. Given the large amounts of gathered data, machine learning techniques can be used to advance research challenges and promote social benefits such as hazard predictions and the preservation of natural resources. In seismology, machine learning methods have been used since the nineties for seismic event detection, localization, and classification. More recently, deep learning architectures have been applied for modeling larger amounts of seismic data to provide fast and accurate event detection and classification solutions incorporated into real-time analysis systems. Rabin and Bregman review the noticeable research trends and

developments in the field and discuss advantages, drawbacks, and future possible research directions.

Chapter "Sentiment Analysis for Social Text" focuses on sentiment analysis, the computational detection, and study of opinions and viewpoints underlying a text span. In social text settings: short, informal, and noisy text spans. The chapter presents an ontology of the field and explores the relevant tasks for social data: lexical-, aspect-, and sentence-level sentiment analysis along with their methods, applications, and resources.

Organizational data mining (ODM) is defined as leveraging data mining (DM) tools and technologies to enhance the organizational decision-making process by transforming data into valuable and actionable knowledge to gain a strategic competitive advantage. Chapter "Human Resources Based Organizational Data Mining (HRODM): Themes, Trends, Focus, Future" presents a literature review of human-resources-based organizational data mining (HRODM). Moreover, this chapter discusses practical implementation tools to assist decision-makers concerning whether and in which format to implement HRODM. A framework is presented that aggregates the findings and clarifies how various HRODM tools influence return on investment (ROI) and how these relationships can be explained.

7.8 Human Factors and Social Issues

The last and final part of this handbook deals with human factors and social issues that should be considered while doing data science. Machine learning models are controlling an increasing number of decisions regarding the daily lives of human beings. Since they now touch on many aspects of our lives, it is crucial to develop ML algorithms that are not only accurate but also objective and fair. Chapter "Algorithmic Fairness" begins by discussing the causes of algorithmic bias and unfairness and the common definitions and measures for fairness. Fairness-enhancing mechanisms are then reviewed and divided into preprocess, in-process, and post-process mechanisms. A comprehensive comparison of the mechanisms is then conducted toward a better understanding of which mechanisms should be used in different scenarios. Finally, Chap. "Algorithmic Fairness" describes the most commonly used fairness-related data sets in this field.

Privacy has become one of the most significant concerns in the digital era, mainly due to the information disclosure enabled by data mining. Privacy- preserving data mining (PPDM) is a collection of methodologies aimed to minimize and control the amount of private information disclosure in data mining processes. Chapter "Privacy-Preserving Data Mining (PPDM)" presents the various approaches to achieve PPDM: anonymization, randomization, cryptography, and privatizing results as well as various common methodologies and techniques used to implement these approaches.

The visual exploration of multidimensional data for knowledge discovery is a long-standing challenge due to the possible loss of information. Chapter "Explainable Machine Learning and Visual Knowledge Discovery" explains the differences

between analytical and visual ML methods and approaches, showing the benefits of visual methods for ML. Next, several methods to visualize ML models are presented, including input-based and structure-based methods accompanied by examples. A major part of the chapter is devoted to the approaches and the theory, to discover interpretable analytical ML models aided by visual methods.

The rapidly developing AI systems and applications still require human involvement in practically all parts of the analytics process. Human decisions are primarily based on visualizations, providing data scientists with details of data properties and the results of analytical procedures. Chapter "Visual Analytics and Human Involvement in Machine Learning" describes the seven steps in the ML process and reviews different visualization techniques that are relevant for the different steps for different types of data, models, and purposes.

Explainable artificial intelligence (XAI) or interpretable machine learning (IML) methods aim to produce more explainable models while maintaining a high level of output accuracy. It lets users better understand, trust, and manage the emerging generation of artificially intelligent systems. Chapter "Explainable Artificial Intelligence (XAI): Motivation, Terminology and Taxonomy" presents various XAI-related concepts of explainability, interpretability, and accuracy, followed by a taxonomy of XAI methods.

Handling Missing Attribute Values

Jerzy W. Grzymala-Busse and Witold J. Grzymala-Busse

1 Introduction

We assume that input data for data mining are presented in a form of a *decision table* (or *data set*) in which *cases* (or *records*) are described by *attributes* (independent variables) and a *decision* (dependent variable). A very simple example of such a table is presented in Table 1, with the attributes *Temperature*, *Headache*, and *Nausea* and with the decision *Flu*. However, many real-life data sets are incomplete, i.e., some attribute values are missing. In Table 1, missing attribute values are denoted by "?"s.

The set of all cases with the same decision value is called a *concept*. For Table 1, case set {1, 2, 4, 8} is a concept of all cases such that the value of *Flu* is *yes*.

There is a variety of reasons why data sets are affected by missing attribute values. Some attribute values are not recorded because they are irrelevant. For example, a doctor was able to diagnose a patient without some medical tests, or a home owner was asked to evaluate the quality of air conditioning, while the home was not equipped with an air conditioner. Such missing attribute values will be called *"do not care" conditions*.

Another reason for missing attribute values is that the attribute value was not placed into the table because it was forgotten or it was placed into the table but later on was mistakenly erased. Sometimes a respondent refuses to answer a question. Such a value, which matters but that is missing, will be called *lost*.

J. W. Grzymala-Busse (✉)
University of Kansas, Lawrence, KS, USA
e-mail: jerzygb@ku.edu; jerzy@ku.edu

W. J. Grzymala-Busse
TouchNet Information Systems, Inc., Lenexa, KS, USA

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_2

Table 1 An example of a data set with missing attribute values

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	high	?	no	yes
2	very_high	yes	yes	yes
3	?	no	no	no
4	high	yes	yes	yes
5	high	?	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	?	yes	?	yes

The problem of missing attribute values is as important for data mining as it is for statistical reasoning. In both disciplines, there are methods to deal with missing attribute values. Some theoretical properties of data sets with missing attribute values were studied in [28, 36, 37].

In general, methods to handle missing attribute values belong either to *sequential methods* (called also *preprocessing methods*) or to *parallel methods* (methods in which missing attribute values are taken into account during the main process of acquiring knowledge).

Sequential methods include techniques based on deleting cases with missing attribute values, replacing a missing attribute value by the most common value of that attribute, assigning all possible values to the missing attribute value, replacing a missing attribute value by the mean for numerical attributes, assigning to a missing attribute value the corresponding value taken from the closest fit case, or replacing a missing attribute value by a new value, computed from a new data set, considering the original attribute as a decision.

The second group of methods to handle missing attribute values, in which missing attribute values are taken into account during the main process of acquiring knowledge, is represented, for example, by a modification of the LEM2 (Learning from Examples Module, version 2) rule induction algorithm in which rules are induced from the original data set, with missing attribute values considered to be "do not care" conditions or lost values. C4.5 [43] approach to missing attribute values is another example of a method from this group. C4.5 induces a decision tree during tree generation, splitting cases with missing attribute values into fractions and adding these fractions to new case subsets. A method of *surrogate splits* to handle missing attribute values was introduced in CART [3], yet another system to induce decision trees. Other methods of handling missing attribute values while generating decision trees were presented in [2, 4].

In statistics, *pairwise deletion* [1, 38, 39] is used to evaluate statistical parameters from available information.

In this chapter, we assume that the main process is rule induction. Additionally, for the rest of the chapter, we will assume that all decision values are known, i.e., specified. Also, we will assume that for each case at least one attribute value is known.

2 Sequential Methods

In sequential methods to handle missing attribute values, original incomplete data sets, with missing attribute values, are converted into complete data sets, and then the main process, e.g., rule induction, is conducted.

2.1 Deleting Cases with Missing Attribute Values

This method is based on ignoring cases with missing attribute values. It is also called *listwise deletion* (or *casewise deletion*, or *complete case analysis*) in statistics. All cases with missing attribute values are deleted from the data set. For the example presented in Table 1, a new table, presented in Table 2, is created as a result of this method.

Obviously, a lot of information is missing in Table 2. However, there are some reasons [1, 38] to consider it a feasible method.

2.2 The Most Common Value of an Attribute

In this method, one of the simplest methods to handle missing attribute values, such values are replaced by the most common value of the attribute. In different words, a missing attribute value is replaced by the most probable known attribute value, where such probabilities are represented by relative frequencies of corresponding attribute values. This method of handling missing attribute values is implemented, e.g., in CN2 [6]. In our example from Table 1, a result of using this method is presented in Table 3.

For case 1, the value of *Headache* in Table 3 is *yes* since in Table 1 the attribute *Headache* has four values *yes* and two values *no*. Similarly, for case 3, the value of *Temperature* in Table 3 is *high* since the attribute *Temperature* has the value *very_high* once, *normal* twice, and *high* three times.

Table 2 Data set with deleted cases with missing attribute values

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	very_high	yes	yes	yes
2	high	yes	yes	yes
3	normal	yes	no	no
4	normal	no	yes	no

Table 3 Data set with missing attribute values replaced by the most common values

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	high	yes	no	yes
2	very_high	yes	yes	yes
3	high	no	no	no
4	high	yes	yes	yes
5	high	yes	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	high	yes	yes	yes

Table 4 Data set with missing attribute values replaced by the most common value of the attribute restricted to a concept

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	high	yes	no	yes
2	very_high	yes	yes	yes
3	normal	no	no	no
4	high	yes	yes	yes
5	high	no	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	high	yes	yes	yes

2.3 The Most Common Value of an Attribute Restricted to a Concept

A modification of the method of replacing missing attribute values by the most common value is a method in which the most common value of the attribute restricted to the concept is used instead of the most common value for all cases. Such a concept is the same concept that contains the case with missing attribute value.

Let us say that attribute a has missing attribute value for case x from concept C and that the value of a for x is missing. This missing attribute value is exchanged by the known attribute value for which the conditional probability of a for case x given C is the largest. This method was implemented, e.g., in ASSISTANT [43]. In our example from Table 1, a result of using this method is presented in Table 4.

For example, in Table 1, case 1 belongs to the concept {1, 2, 4, 8}, all known values of *Headache*, restricted to {1, 2, 4, 8}, are *yes*, so the missing attribute value is replaced by *yes*. On the other hand, in Table 1, case 3 belongs to the concept {3, 5, 6, 7}, and the value of *Temperature* is missing. The known values of *Temperature*, restricted to {3, 5, 6, 7}, are: *high* (once) and *normal* (twice), so the missing attribute value is exchanged by *normal*.

2.4 Assigning All Possible Attribute Values to a Missing Attribute Value

This approach to missing attribute values was presented for the first time in [15] and implemented in LERS. Every case with missing attribute values is replaced by the set of cases in which every missing attribute value is replaced by all possible known values. In the example from Table 1, a result of using this method is presented in Table 5.

In the example of Table 1, the first case from Table 1, with the missing attribute value for attribute *Headache*, is replaced by two cases, 1^{i} and 1^{ii}, where case 1^{i} has value *yes* for attribute *Headache*, and case 1^{ii} has values *no* for the same attribute, since attribute *Headache* has two possible known values, *yes* and *no*. Case 3 from Table 1, with the missing attribute value for the attribute *Temperature*, is replaced by three cases, 3^{i}, 3^{ii}, and 3^{iii}, with values *high*, *very_high*, and *normal*, since the attribute *Temperature* has three possible known values, *high*, *very_high*, and *normal*, respectively. Note that due to this method, the new table, such as Table 5, may be inconsistent. In Table 5, case 1^{ii} conflicts with case 3^{i}, case 4 conflicts with case 5^{i}, etc. However, rule sets may be induced from inconsistent data sets using standard rough set techniques, see, e.g., [14, 15, 16, 17, 18].

Table 5 Data set in which all possible values are assigned to missing attribute values

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1^{i}	high	yes	no	yes
1^{ii}	high	no	no	yes
2	very_high	yes	yes	yes
3^{i}	high	no	no	no
3^{ii}	very_high	no	no	no
3^{iii}	normal	no	no	no
4	high	yes	yes	yes
5^{i}	high	yes	yes	no
5^{ii}	high	no	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8^{i}	high	yes	yes	yes
8^{ii}	high	yes	no	yes
8^{iii}	very_high	yes	yes	yes
8^{iv}	very_high	yes	no	yes
8^{v}	normal	yes	yes	yes
8^{vi}	normal	yes	no	yes

Table 6 Data set in which all possible values, restricted to the concept, are assigned to missing attribute values

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	high	yes	no	yes
2	very_high	yes	yes	yes
3^{i}	normal	no	no	no
3^{ii}	high	no	no	no
4	high	yes	yes	yes
5^{i}	high	yes	yes	no
5^{ii}	high	no	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8^{i}	high	yes	yes	yes
8^{ii}	high	yes	no	yes
8^{iii}	very_high	yes	yes	yes
8^{iv}	very_high	yes	no	yes

2.5 *Assigning All Possible Attribute Values Restricted to a Concept*

This method was described, e.g., in [25]. Here, every case with missing attribute values is replaced by the set of cases in which every attribute a with the missing attribute value has its every possible known value restricted to the concept to which the case belongs. In the example from Table 1, a result of using this method is presented in Table 6.

In the example of Table 1, the first case from Table 1, with the missing attribute value for attribute *Headache*, is replaced by one with value *yes* for attribute *Headache*, since attribute *Headache*, restricted to the concept {1, 2, 4, 8}, has one possible known value, *yes*. Case 3 from Table 1, with the missing attribute value for the attribute *Temperature*, is replaced by two cases, 3^{i} and 3^{ii}, with values *high* and *very_high*, since the attribute *Temperature*, restricted to the concept {3, 5, 6, 7}, has two possible known values, *normal* and *high*, respectively. Again, due to this method, the new table, such as Table 6, may be inconsistent. In Table 6, case 4 conflicts with case 5^{i}, etc.

2.6 *Replacing Missing Attribute Values by the Attribute Mean*

This method is used for data sets with numerical attributes. An example of such a data set is presented in Table 7.

In this method, every missing attribute value for a numerical attribute is replaced by the arithmetic mean of known attribute values. In Table 7, the mean of known attribute values for Temperature is 99.2; hence, all missing attribute values for

Table 7 An example of a data set with a numerical attribute

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	100.2	?	no	yes
2	102.6	yes	yes	yes
3	?	no	no	no
4	99.6	yes	yes	yes
5	99.8	?	yes	no
6	96.4	yes	no	no
7	96.6	no	yes	no
8	?	yes	?	yes

Table 8 Data set in which missing attribute values are replaced by the attribute mean and the most common value

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	100.2	yes	no	yes
2	102.6	yes	yes	yes
3	99.2	no	no	no
4	99.6	yes	yes	yes
5	99.8	yes	yes	no
6	96.4	yes	no	no
7	96.6	no	yes	no
8	99.2	yes	yes	yes

Temperature should be replaced by 99.2. The table with missing attribute values replaced by the mean is presented in Table 8. For symbolic attributes *Headache* and *Nausea*, missing attribute values were replaced using the most common value of the attribute.

2.7 Replacing Missing Attribute Values by the Attribute Mean Restricted to a Concept

Similarly as in the previous method, this method is restricted to numerical attributes. A missing attribute value of a numerical attribute is replaced by the arithmetic mean of all known values of the attribute restricted to the concept. For example from Table 7, case 3 has missing attribute value for *Temperature*. Case 3 belongs to the concept {3, 5, 6, 7}. The arithmetic mean of known values of *Temperature* restricted to the concept, i.e., 99.8, 96.4, and 96.6, is 97.6, so the missing attribute value is replaced by 97.6. On the other hand, case 8 belongs to the concept {1, 2, 4, 8}, the arithmetic mean of 100.2, 102.6, and 99.6 is 100.8, so the missing attribute value for case 8 should be replaced by 100.8. The table with missing attribute values replaced by the mean restricted to the concept is presented in Table 9. For symbolic

Table 9 Data set in which missing attribute values are replaced by the attribute mean and the most common value, both restricted to the concept

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	100.2	yes	no	yes
2	102.6	yes	yes	yes
3	97.6	no	no	no
4	99.6	yes	yes	yes
5	99.8	no	yes	no
6	96.4	yes	no	no
7	96.6	no	yes	no
8	100.8	yes	yes	yes

attributes *Headache* and *Nausea*, missing attribute values were replaced using the most common value of the attribute restricted to the concept.

2.8 Global Closest Fit

The global closes fit method [24] is based on replacing a missing attribute value by the known value in another case that resembles as much as possible the case with the missing attribute value. In searching for the closest fit case, we compare two vectors of attribute values, one vector corresponds to the case with a missing attribute value, and the other vector is a candidate for the closest fit. The search is conducted for all cases, hence the name global closest fit. For each case, a distance is computed, and the case for which the distance is the smallest is the closest fitting case that is used to determine the missing attribute value. Let x and y be two cases. The distance between cases x and y is computed as follows:

$$\text{distance}(x, y) = \sum_{i=1}^{n} \text{distance}(x_i, y_i),$$

where

$$\text{distance}(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \text{ ,} \\ 1 & \text{if } x \text{ and } y \text{ are symbolic and } x_i \neq y_i, \\ & \text{or } x_i = ? \text{ or } y_i = ?, \\ \frac{|x_i - y_i|}{r} & \text{if } x_i \text{ and } y_i \text{ are numbers and } x_i \neq y_i, \end{cases}$$

where r is the difference between the maximum and minimum of the known values of the numerical attribute with a missing value. If there is a tie for two cases with the same distance, a kind of heuristics is necessary, for example, select the first case. In general, using the global closest fit method may result in data sets in which some missing attribute values are not replaced by known values. Additional iterations of

Table 10 Distance (1, x)

d(1, 2)	d(1, 3)	d(1, 4)	d(1, 5)	d(1, 6)	d(1, 7)	d (1, 8)
2.39	2.0	2.10	2.06	1.61	2.58	3.00

Table 11 Data set processed by the global closest fit method

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	100.2	yes	no	yes
2	102.6	yes	yes	yes
3	100.2	no	no	no
4	99.6	yes	yes	yes
5	99.8	yes	yes	no
6	96.4	yes	no	no
7	96.6	no	yes	no
8	102.6	yes	yes	yes

using this method may reduce the number of missing attribute values but may not end up with all missing attribute values being replaced by known attribute values.

For the data set in Table 7, distances between case 1 and all remaining cases are presented in Table 10. For example, the distance $d(1, 2) = \frac{|100.2-102.6|}{|102.6-96.4|} + 1 + 1 = 2.39$. For case 1, the missing attribute value (for attribute *Headache*) should be the value of *Headache* for case 6, i.e., *yes*, since for this case the distance is the smallest. The table with missing attribute values replaced by values computed on the basis of the global closest fit is presented in Table 11. Table 11 is complete. However, in general, some missing attribute values may still be present in such a table. If so, it is recommended to use another method of handling missing attribute values to replace all remaining missing attribute values by some specified attribute values.

2.9 Concept Closest Fit

This method is similar to the global closest fit method. The difference is that the original data set, containing missing attribute values, is first split into smaller data sets, and each smaller data set corresponds to a concept from the original data set. More precisely, every smaller data set is constructed from one of the original concepts, by restricting cases to the concept. For the data set from Table 7, two smaller data sets are created, presented in Tables 12 and 13.

Following the data set split, the same global closest fit method is applied to both tables separately. Eventually, both tables, processed by the global fit method, are merged into the same table. In our example from Table 7, the final, merged table is presented in Table 14.

Table 12 Data set restricted to the concept {1, 2, 4, 8}

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	100.2	?	no	yes
2	102.6	yes	yes	yes
4	99.6	yes	yes	yes
8	?	yes	?	yes

Table 13 Data set restricted to the concept {3, 5, 6, 7}

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
3	?	no	no	no
5	99.8	?	yes	no
6	96.4	yes	no	no
7	96.6	no	yes	no

Table 14 Data set processed by the concept closest fit method

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	100.2	yes	no	yes
2	102.6	yes	yes	yes
3	96.4	no	no	no
4	99.6	yes	yes	yes
5	99.8	no	yes	no
6	96.4	yes	no	no
7	96.6	no	yes	no
8	102.6	yes	yes	yes

2.10 Other Methods

There is a number of other methods to handle missing attribute values. One of them is *event-covering method* [5, 49], based on an interdependency between known and missing attribute values. The interdependency is computed from contingency tables. The outcome of this method is not necessarily a complete data set (with all attribute values known), just like in the case of closest fit methods.

Another method of handling missing attribute values, called D^3RJ, was discussed in [32, 33]. In this method, a data set is decomposed into complete data subsets, rule sets are induced from such data subsets, and finally, these rule sets are merged.

Yet another method of handling missing attribute values was referred to as Shapiro's method in [42], where for each attribute with missing attribute values a new data set is created, such attributes take place of the decision and vice versa, and the decision becomes one of the attributes. From such a table, missing attribute values are learned using either a rule set or decision tree techniques. This method, identified as a *chase* algorithm, was also discussed in [11, 12].

Learning missing attribute values from summary constraints was reported in [50]. Yet another approach to handling missing attribute values was presented in [13].

There is a number of statistical methods of handling missing attribute values, usually known under the name of *imputation* [1, 38, 44], such as maximum likelihood and the EM algorithm. Recently, *multiple imputation* gained popularity. It is a Monte Carlo method of handling missing attribute values in which missing attribute values are replaced by many plausible values, then many complete data sets are analyzed, and the results are combined.

3 Parallel Methods

In this section, we will concentrate on handling missing attribute values in parallel with rule induction. We will distinguish two types of missing attribute values: *lost* and *do not care* conditions (for respective interpretation, see Introduction). First, we will introduce some useful ideas, such as blocks of attribute-value pairs, characteristic sets, characteristic relations, and lower and upper approximations. Later, we will explain how to induce rules using the same blocks of attribute-value pairs that were used to compute lower and upper approximations. Input data sets are not preprocessed the same way as in sequential methods; instead, the rule learning algorithm is modified to learn rules directly from the original, incomplete data sets.

3.1 Blocks of Attribute-Value Pairs and Characteristic Sets

In this subsection, we will quote some basic ideas of the rough set theory. Any decision table defines a function ρ that maps the direct product of the set U of all cases and the set A of all attributes into the set of all values. For example, in Table 1, $\rho(1, Temperature) = high$. In this section, we will assume that all missing attribute values are denoted either by "?" or by "*," lost values will be denoted by "?," and "do not care" conditions will be denoted by "*." Thus, we assume that all missing attribute values from Table 1 are lost. On the other hand, all attribute values from Table 15 are "do not care" conditions.

Let (a, v) be an attribute-value pair. For complete decision tables, a block of (a, v), denoted by $[(a, v)]$, is the set of all cases x for which $\rho(x, a) = v$. For incomplete decision tables, the definition of a block of an attribute-value pair is modified. If for an attribute a there exists a case x such that $\rho(x, a) =?$, i.e., the corresponding value is lost, then the case x is not included in any block $[(a, v)]$ for every value v of attribute a. If for an attribute a there exists a case x such that the corresponding value is a "do not care" condition, i.e., $\rho(x, a) = *$, then the corresponding case x should be included in blocks $[(a, v)]$ for all known values v of attribute a. This modification of the attribute-value pair block definition is consistent

Table 15 An example of a data set with "do not care" conditions

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	high	*	no	yes
2	very_high	yes	yes	yes
3	*	no	no	no
4	high	yes	yes	yes
5	high	*	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	*	yes	*	yes

with the interpretation of missing attribute values, lost and "do not care" conditions. Thus, for Table 1

[(Temperature, high)] = {1, 4, 5},
[(Temperature, very_high)] = {2},
[(Temperature, normal)] = {6, 7},
[(Headache, yes)] = {2, 4, 6, 8},
[(Headache, no)] = {3, 7},
[(Nausea, no)] = {1, 3, 6},
[(Nausea, yes)] = {2, 4, 5, 7},

and for Table 15

[(Temperature, high)] = {1, 3, 4, 5, 8},
[(Temperature, very_high)] = {2, 3, 8},
[(Temperature, normal)] = {3, 6, 7, 8},
[(Headache, yes)] = {1, 2, 4, 5, 6, 8},
[(Headache, no)] = {1, 3, 5, 7},
[(Nausea, no)] = {1, 3, 6, 8},
[(Nausea, yes)] = {2, 4, 5, 7, 8}.

The *characteristic set* $K_B(x)$ is the intersection of blocks of attribute-value pairs (a, v) for all attributes a from B for which $\rho(x, a)$ is known and $\rho(x, a) = v$. For Table 1 and $B = A$,

$K_A(1) = \{1, 4, 5\} \cap \{1, 3, 6\} = \{1\}$,
$K_A(2) = \{2\} \cap \{2, 4, 6, 8\} \cap \{2, 4, 5, 7\} = \{2\}$,
$K_A(3) = \{3, 7\} \cap \{1, 3, 6\} = \{3\}$,
$K_A(4) = \{1, 4, 5\} \cap \{2, 4, 6, 8\} \cap \{2, 4, 5, 7\} = \{4\}$,
$K_A(5) = \{1, 4, 5\} \cap \{2, 4, 5, 7\} = \{4, 5\}$,
$K_A(6) = \{6, 7\} \cap \{2, 4, 6, 8\} \cap \{1, 3, 6\} = \{6\}$,
$K_A(7) = \{6, 7\} \cap \{3, 7\} \cap \{2, 4, 5, 7\} = \{7\}$, and
$K_A(8) = \{2, 4, 6, 8\}$.

and for Table 15 and $B = A$,

$K_A(1) = \{1, 3, 4, 5, 8\} \cap \{1, 3, 6, 8\} = \{1, 3, 8\}$,
$K_A(2) = \{2, 3, 8\} \cap \{1, 2, 4, 5, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{2, 8\}$,
$K_A(3) = \{1, 3, 5, 7\} \cap \{1, 3, 6, 8\} = \{1, 3\}$,
$K_A(4) = \{1, 3, 4, 5, 8\} \cap \{1, 2, 4, 5, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 5, 8\}$,
$K_A(5) = \{1, 3, 4, 5, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 5, 8\}$,
$K_A(6) = \{3, 6, 7, 8\} \cap \{1, 2, 4, 5, 6, 8\} \cap \{1, 3, 6, 8\} = \{6, 8\}$,
$K_A(7) = \{3, 6, 7, 8\} \cap \{1, 3, 5, 7\} \cap \{2, 4, 5, 7, 8\} = \{7\}$, and
$K_A(8) = \{1, 2, 4, 5, 6, 8\}$.

The characteristic set $K_B(x)$ may be interpreted as the smallest set of cases that are indistinguishable from x using all attributes from B, using a given interpretation of missing attribute values. Thus, $K_A(x)$ is the set of all cases that cannot be distinguished from x using all attributes. For further properties of characteristic sets, see [19, 21, 20, 22]. Incomplete decision tables in which all attribute values are lost, from the viewpoint of rough set theory, were studied for the first time in [27], where two algorithms for rule induction, modified to handle lost attribute values, were presented. This approach was studied later in [45, 46, 47].

Incomplete decision tables in which all missing attribute values are "do not care" conditions, from the view point of rough set theory, were studied for the first time in [15], where a method for rule induction was introduced in which each missing attribute value was replaced by all values from the domain of the attribute. Originally, such values were replaced by all values from the entire domain of the attribute, later, by attribute values restricted to the same concept to which a case with a missing attribute value belongs. Such incomplete decision tables, with all missing attribute values being "do not care" conditions, were also studied in [29, 30]. Both approaches to missing attribute values were generalized in [19, 21, 20, 22].

3.2 Lower and Upper Approximations

Any finite union of characteristic sets of B is called a *B-definable* set. The lower approximation of the concept X is the largest definable set that is contained in X, and the upper approximation of X is the smallest definable set that contains X. In general, for incompletely specified decision tables, lower and upper approximations may be defined in a few different ways [19, 21, 20, 22]. Here we will quote the most useful definition of lower and upper approximations from the view point of data mining. A *concept* B-lower approximation of the concept X is defined as follows:

$$\underline{B}X = \cup\{K_B(x) | x \in X, K_B(x) \subseteq X\}.$$

A concept B-upper approximation of the concept X is defined as follows:

$$\overline{B}X = \cup\{K_B(x) | x \in X, K_B(x) \cap X \neq \emptyset\} = \cup\{K_B(x) | x \in X\}.$$

For the decision table presented in Table 1, the concept A-lower and A-upper approximations are

$$\underline{A}\{1, 2, 4, 8\} = \{1, 2, 4\},$$

$$\underline{A}\{3, 5, 6, 7\} = \{3, 6, 7\},$$

$$\overline{A}\{1, 2, 4, 8\} = \{1, 2, 4, 6, 8\},$$

$$\overline{A}\{3, 5, 6, 7\} = \{3, 4, 5, 6, 7\},$$

and for the decision table from Table 15, the concept A-lower and A-upper approximations are

$$\underline{A}\{1, 2, 4, 8\} = \{2, 8\},$$

$$\underline{A}\{3, 5, 6, 7\} = \{7\},$$

$$\overline{A}\{1, 2, 4, 8\} = \{1, 2, 3, 4, 5, 6, 8\},$$

$$\overline{A}\{3, 5, 6, 7\} = \{1, 3, 4, 5, 6, 7, 8\}.$$

3.3 Rule Induction—MLEM2

The MLEM2 rule induction algorithm is a modified version of the algorithm LEM2, see Chap. "Generative Adversarial Networks" in this volume. Rules induced from the lower approximation of the concept *certainly* describe the concept, so they are called *certain*. On the other hand, rules induced from the upper approximation of the concept describe the concept only *possibly* (or *plausibly*), so they are called *possible* [14]. MLEM2 may induce both certain and possible rules from a decision table with some missing attribute values being lost and some missing attribute values being "do not care" conditions, while some attributes may be numerical. For rule induction from decision tables with numerical attributes, see [21]. MLEM2 handles missing attribute values by computing (in a different way than in LEM2) blocks of attribute-value pairs, and then characteristic sets and lower and upper approximations. All these definitions are modified according to the two previous subsections, and the algorithm itself remains the same.

Rule sets in the LERS format (every rule is equipped with three numbers, the total number of attribute-value pairs on the left-hand side of the rule, the total number of examples correctly classified by the rule during training, and the total number of training cases matching the left-hand side of the rule), induced from the decision table presented in Table 1 are:
certain rule set:

2, 1, 1
(Temperature, high) & (Nausea, no) -> (Flu, yes)
2, 2, 2
(Headache, yes) & (Nausea, yes) -> (Flu, yes)
1, 2, 2
(Temperature, normal) -> (Flu, no)
1, 2, 2
(Headache, no) -> (Flu, no)

and possible rule set:

1, 3, 4
(Headache, yes) -> (Flu, yes)
2, 1, 1
(Temperature, high) & (Nausea, no) -> (Flu, yes)
2, 1, 2
(Temperature, high) & (Nausea, yes) -> (Flu, no)
1, 2, 2
(Temperature, normal) -> (Flu, no)
1, 2, 2
(Headache, no) -> (Flu, no)

Rule sets induced from the decision table presented in Table 15 are:
certain rule set:

2, 2, 2
(Temperature, very_high) & (Nausea, yes) -> (Flu, yes)
3, 1, 1
(Temperature, normal) & (Headache, no) & (Nausea, yes) -> (Flu, no)

and possible rule set:

1, 4, 6
(Headache, yes) -> (Flu, yes)
1, 2, 3
(Temperature, very_high) -> (Flu, yes)
1, 2, 5
(Temperature, high) -> (Flu, no)
1, 3, 4
(Temperature, normal) -> (Flu, no)

3.4 Other Approaches to Missing Attribute Values

Through this section, we assumed that the incomplete decision tables may only consist of lost values or “do not care” conditions. Note that the MLEM2 algorithm

is able to handle not only these two types of tables but also decision tables with a mixture of these two cases, i.e., tables with some lost attribute values and with other missing attribute values being "do not care" conditions. Furthermore, other interpretations of missing attribute values are possible as well, see [19, 21].

Both lower and upper approximations for incomplete data sets were generalized to probabilistic approximations in [23] and further studied, e.g., in [8, 9, 10]. Probabilistic approximations are associated with a parameter interpreted as a probability; when this parameter is equal to one, the probabilistic approximation is reduced to the lower approximation; when the parameter is a small, positive number, the probabilistic approximation becomes the upper approximation.

An idea of maximal consistent blocks was introduced in [34, 35] for data sets with "do not care" conditions. This idea was generalized for arbitrary incomplete data sets in [7]. Yet other approaches to incomplete data sets were presented in [40, 41].

4 Conclusions

In general, there is no best, universal method of handling missing attribute values. On the basis of existing research on comparison of such methods [25, 26, 31], we may conclude that for every specific data set the best method of handling missing attribute values should be chosen individually, using as the criterion of optimality the arithmetic mean of many multi-fold cross-validation experiments [48]. Similar conclusions may be drawn for decision tree generation [42].

References

1. Allison, P.D.: Missing Data. Sage Publications, Thousand Oaks, CA (2002)
2. Brazdil, P., Bruha, I.: Processing unknown attribute values by ID3. In: Proceedings of the 4-th International Conference on Computing and Information. pp. 227–230 (1992)
3. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth & Brooks, Monterey, CA (1984)
4. Bruha, I.: Meta-learner for unknown attribute values processing: Dealing with inconsistency of meta-databases. Journal of Intelligent Information Systems **22**, 71–87 (2004)
5. Chiu, D.K., Wong, A.K.C.: Synthesizing knowledge: A cluster analysis approach using event-covering. IEEE Transactions Syst., Man, and Cybernet **16**, 251–259 (1986)
6. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning **3**, 261–283 (1989)
7. Clark, P.G., Gao, C., Grzymala-Busse, J.W., Mroczek, T.: Characteristic sets and generalized maximal consistent blocks in mining incomplete data. Information Sciences **453**, 66–79 (2018)
8. Clark, P.G., Grzymala-Busse, J.W.: Experiments on probabilistic approximations. In: Proceedings of the 2011 IEEE International Conference on Granular Computing. pp. 144–149 (2011)

9. Clark, P.G., Grzymala-Busse, J.W.: Experiments on rule induction from incomplete data using three probabilistic approximations. In: Proceedings of the 2012 IEEE International Conference on Granular Computing. pp. 90–95 (2012)
10. Clark, P.G., Grzymala-Busse, J.W., Rzasa, W.: Mining incomplete data with singleton, subset and concept approximations. Information Sciences **280**, 368–384 (2014)
11. Dardzinska, A., Ras, Z.W.: Chasing unknown values in incomplete information systems. In: Workshop Notes, Foundations and New Directions of Data Mining, in conjunction with the 3rd International Conference on Data Mining. pp. 24–30 (2003)
12. Dardzinska, A., Ras, Z.W.: On rule discovery from incomplete information systems. In: Workshop Notes, Foundations and New Directions of Data Mining, in conjunction with the 3-rd International Conference on Data Mining. pp. 24–30 (2003)
13. Greco, S., Matarazzo, B., Slowinski, R.: Dealing with missing data in rough set analysis of multi-attribute and multi-criteria decision problems. In: Zanakis, H., Doukidis, G., Zopounidised, Z. (eds.) Decision Making: Recent Developments and Worldwide Applications, pp. 295–316. Kluwer Academic Publishers, Dordrecht, Boston, London (2000)
14. Grzymala-Busse, J.W.: Knowledge acquisition under uncertainty—A rough set approach. Journal of Intelligent & Robotic Systems **1**, 3–16 (1988)
15. Grzymala-Busse, J.W.: On the unknown attribute values in learning from examples. In: Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems. pp. 368–377 (1991)
16. Grzymala-Busse, J.W.: LERS—a system for learning from examples based on rough sets. In: Slowinski, R. (ed.) Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory, pp. 3–18. Kluwer Academic Publishers, Dordrecht, Boston, London (1992)
17. Grzymala-Busse, J.W.: A new version of the rule induction system LERS. Fundamenta Informaticae **31**, 27–39 (1997)
18. Grzymala-Busse, J.W.: MLEM2: A new algorithm for rule induction from imperfect data. In: Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. pp. 243–250 (2002)
19. Grzymala-Busse, J.W.: Rough set strategies to data with missing attribute values. In: Notes of the Workshop on Foundations and New Directions of Data Mining, in conjunction with the Third International Conference on Data Mining. pp. 56–63 (2003)
20. Grzymala-Busse, J.W.: Characteristic relations for incomplete data: A generalization of the indiscernibility relation. In: Proceedings of the Fourth International Conference on Rough Sets and Current Trends in Computing. pp. 244–253 (2004)
21. Grzymala-Busse, J.W.: Data with missing attribute values: Generalization of indiscernibility relation and rule induction. Transactions on Rough Sets **1**, 78–95 (2004)
22. Grzymala-Busse, J.W.: Rough set approach to incomplete data. In: Proceedings of the ICAISC, the Seventh International Conference on Artificial Intelligence and Soft Computing. pp. 50–55 (2004)
23. Grzymala-Busse, J.W.: Generalized parameterized approximations. In: Proceedings of the 6th International Conference on Rough Sets and Knowledge Technology. pp. 136–145 (2011)
24. Grzymala-Busse, J.W., Grzymala-Busse, W.J., Goodwin, L.K.: A comparison of three closest fit approaches to missing attribute values in preterm birth data. International Journal of Intelligent Systems **17**(2), 125–134 (2002)
25. Grzymala-Busse, J.W., Hu, M.: A comparison of several approaches to missing attribute values in data mining. In: Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing. pp. 340–347 (2000)
26. Grzymala-Busse, J.W., Siddhaye, S.: Rough set approaches to rule induction from incomplete data. In: Proceedings of the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. pp. 923–930 (2004)
27. Grzymala-Busse, J.W., Wang, A.Y.: Modified algorithms LEM1 and LEM2 for rule induction from data with missing attribute values. In: Proceedings of the 5th International Workshop

on Rough Sets and Soft Computing in conjunction with the Third Joint Conference on Information Sciences. pp. 69–72 (1997)
28. Imielinski, T., Lipski, W.J.: Incomplete information in relational databases. Journal of the ACM **31**, 761–791 (1984)
29. Kryszkiewicz, M.: Rough set approach to incomplete information systems. In: Proceedings of the Second Annual Joint Conference on Information Sciences. pp. 194–197 (1995)
30. Kryszkiewicz, M.: Rules in incomplete information systems. Information Sciences **113**(3–4), 271–292 (1999)
31. Lakshminarayan, K., A., H.S., Samad, T.: Imputation of missing data in industrial databases. Applied Intelligence **11**, 259–275 (1999)
32. Latkowski, R.: On decomposition for incomplete data. Fundamenta Informaticae **54**, 1–16 (2003)
33. Latkowski, R., Mikolajczyk, M.: Data decomposition and decision rule joining for classification of data with missing values. In: Proceedings of the Fourth International Conference on Rough Sets and Current Trends in Computing. pp. 254–263 (2004)
34. Leung, Y., Li, D.: Maximal consistent block technique for rule acquisition in incomplete information systems. Information Sciences **153**, 85–106 (2003)
35. Leung, Y., Wu, W., Zhang, W.: Knowledge acquisition in incomplete information systems: A rough set approach. European Journal of Operational Research **168**, 164–180 (2006)
36. Lipski, W.J.: On semantic issues connected with incomplete information databases. ACM Transactions on Database Systems **4**, 262–296 (1979)
37. Lipski, W.J.: On databases with incomplete information. Journal of the ACM **28**, 41–70 (1981)
38. Little, R.J.A., Rubin, D.B.: Statistical Analysis with Missing Data, Second Edition. J. Wiley & Sons, Inc., Hoboken, NJ (2002)
39. McKnight, P.E., McKnight, K.M., Sidani, S., Figueredo, A.J.: Missing Data. A Gentle Introduction. The Guilford Press, New York, NY (2007)
40. Meng, Z., Shi, Z.: Extended rough set-based attribute reduction in inconsistent incomplete decision systems. Information Sciences **204**, 44–69 (2012)
41. Nakata, M., Sakai, H.: Applying rough sets to information tables containing missing values. In: Proceedings of the 39th International Symposium on Multiple-Valued Logic. pp. 286–291 (2009)
42. Quinlan, J.R.: Unknown attribute values in induction. In: Proceedings of the 6th Int. Workshop on Machine Learning. pp. 164–168 (1989)
43. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA (1993)
44. Schafer, J.L.: Analysis of Incomplete Multivariate Data. Chapman and Hall, London (1997)
45. Stefanowski, J.: Algorithms of Decision Rule Induction in Data Mining. Poznan University of Technology Press, Poznan, Poland (2001)
46. Stefanowski, J., Tsoukias, A.: On the extension of rough sets under incomplete information. In: Proceedings of the RSFDGrC'1999, 7th International Workshop on New Directions in Rough Sets, Data Mining, and Granular-Soft Computing. pp. 73–81 (1999)
47. Stefanowski, J., Tsoukias, A.: Incomplete information tables and rough classification. Computational Intelligence **17**(3), 545–566 (2001)
48. Weiss, S., Kulikowski, C.A.: Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems. Morgan Kaufmann Publ., San Mateo, CA (1991)
49. Wong, A.K.C., Chiu, D.K.Y.: Synthesizing statistical knowledge from incomplete mixed-mode data. IEEE Transactions on Pattern Analysis and Machine Intelligence **9**, 796–805 (1987)
50. Wu, X., Barbara, D.: Learning missing values from summary constraints. ACM SIGKDD Explorations Newsletter **4**, 21–30 (2002)

Data Integration Process Automation Using Machine Learning: Issues and Solution

Kartick Chandra Mondal and Swati Saha

1 Introduction

Data from relevant sources are integrated into data warehouse [17] and mainly used for analytic and reporting purposes. ETL (Extract, Transform, and Load) extracts the data from heterogeneous data sources, cleans and transforms the data, and finally loads it into the data warehouse. The ETL [32] process is complex and consumes a major amount of time, cost, and complexity overhead of any DWH. Operational systems and external systems are typical data sources of DWH. The organization's operational systems include enterprise resource planning (ERP) system, on-line transaction processing (OLTP) system, customer relationship management systems (CRM), etc. External sources can be open to data services or other services. Moreover, some data sources are unstructured or semi-structured like various kinds of documents, spreadsheets, texts, images, or Web pages. In the traditional batch processing ETL system, the DWH is refreshed with the data coming from various data sources weekly or daily basis [34]. These activities are generally performed at night during the warehouse downtime to avoid any unwanted interference.

The traditional data warehouse cannot support continuous integration of data; hence, it does not contain real-time data. In the current business environment, the way of accessing an organization's data is rapidly changing. They want to access reports based on real-time data for taking an immediate decision. Many industries such as stock exchange, air traffic control, e-commerce, telecommunication, etc. need to access information rapidly and react immediately based on real-time

K. C. Mondal (✉)
Department of Information Technology, Jadavpur University, Kolkata, India
e-mail: kartickjgec@gmail.com

S. Saha
Tata Consultancy Services, Kolkata, India
e-mail: swati.saha@tcs.com

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_3

information. If the data warehouse is not updated with real-time data, bad decisions can be made. Besides, the volume of data for analysis is becoming very high. Hence, the demand is for continuous integration of data in DWH so that the window time for loading DWH can be shortened. Therefore, the main focus for business intelligence (BI) lies in the DWH and the ETL process for supporting continuous data flow [23, 30] and decreasing downtime. Detecting any changes in data sources and promptly propagating the changes into DWH are the major challenges.

For near real-time ETL process, some well-known techniques of extraction can be considered [3, 10]. They are triggers, timestamping, enterprise application integration (EAI) middleware, log sniffing, snapshot differential [24, 33], etc. All the above-mentioned options have their own limitation. Here we will explore if there is any other option that can track changes and automatically initiate the process of loading data into DWH. Data need to be cleansed and processed [19] before loading in DWH to improve the quality of data. The fixing of data quality issues [5] is a continuous procedure. The main focus of this work is how to automate the ETL process that can manage the growing volume and a variety of data with better quality. Also, the proposed approach describes how various machine learning approaches can be leveraged in this automation.

The paper is organized in the following way. Section 2 briefly discusses some notable related work in ETL automation and near real-time ETL domain. Some important real-life case studies in different business domains regarding data integration scenarios are investigated in Sect. 3, which showcases the necessity of modernizing data warehouse. An architecture is designed in Sect. 4 concerning overall ETL process automation. The solution approach to case study problems is discussed in Sect. 5. Finally, Sect. 6 concludes the work with brief summary followed by targeted future scope.

2 Related Work

Here, we are mainly focusing on the automated ETL process that supports continuous data integration. It minimizes the latency between data changes in the source system and refreshes the changes in data warehouse. Some research work related to the ETL process is discussed in this section. Research related to ETL is mainly done on modeling and designing at conceptual [7], logical [35], and physical level [31]. [27] describe a semantic Web-based ETL designed with a high level of automation. SysML-based conceptual ETL process modeling has been designed in [7]. An ETL process can be designed using the model-driven approach [6, 21]. This can also automatically generate code from the conceptual model. Embley et al. [14] proposed an ontology-based conceptual model for automatic data extraction.

A model-driven framework using BPMN language is practiced in [2]. The model-to-text transformation can automatically produce code suitable for any ETL commercial tool. An empirical analysis of such programmable ETL tools has been done in [8]. Automatic data loading [11] into the warehouse is done followed by

any business events from any application. An automated architecture is designed to optimize ETL throughput in the article [29]. Vassiliadis et al. in [33] first discussed near real-time ETL process. The various possible technical solutions to fulfill the real-time ETL demands are discussed here. Log-based change data capture methodology is projected by Zhou in [36]. Log contents within two timestamps are compared to track the changes in the source side. Naeem et al. [22] proposed an event-based ETL architecture. A layered approach has been taken for transforming data. Different approaches are taken for master data and transactional data. Master data are not extracted in each transaction. Master data and transactional data are identified based on fields attached to the incoming messages. Transaction data are followed by the required enrichment process, and master data are directed to the master data repository.

The concept of automated ETL process based on scripting technology is proposed by [25]. This chapter describes the usage of scripting technologies to automate the ETL process. This process generates three different types of maps namely source extraction map, transformation map, and finally loading map. These three jobs are processed by any ETL tool using scripting technology. An incremental loading for ETL processing on real-time data integration has been discussed in [9]. Although there are some research done on the real-time ETL processing, a very little work is done on automating the ETL process. ETL automation is still an open problem and gains popularity in recent times.

3 Case Study

In this section, a few case studies are discussed showcasing the necessity of modernizing the data warehouse and the ETL process by automating its processes.

3.1 Manufacturing Industry

In the manufacturing environment, various IT applications are used for different purposes that include planning and scheduling, workforce management, material management, product and process design, production generation and maintenance, sales and marketing, human resource, finance, and so on. The data from different applications need to be analyzed to increase efficiency, improve the process for getting a competitive advantage, and stay ahead of competitors. A major electronics manufacturing company has been set up for data warehouse to integrate data from several applications for a better understanding of the performance of the company [28]. To run a data warehouse successfully, multiple operations need to be performed under correct conditions and at the correct time and sequence. Also, significant effort is needed for coordination among several teams including different application teams, database teams, and operation teams as shown in Fig. 1. For

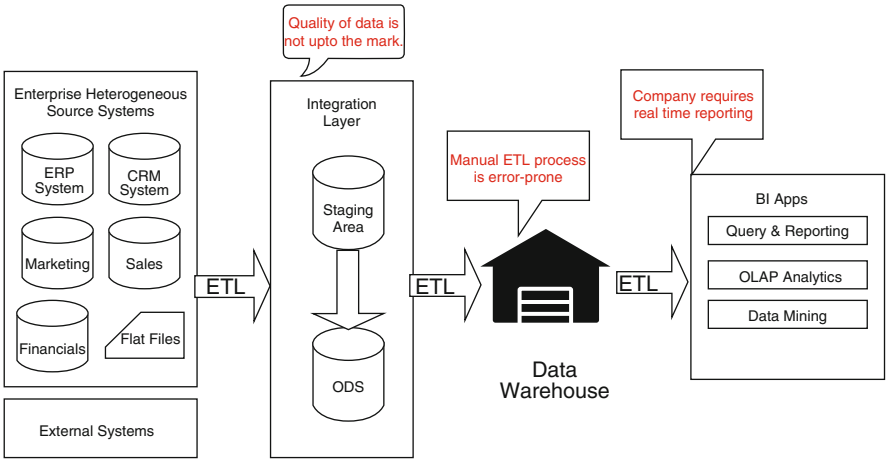


Fig. 1 Challenges in manufacturing industry

this company, all these operations are manual, which magnifies the risk of human-induced errors. There is a high probability to miss one step in the process or execute a step at the wrong time that can produce wrong data or result in a significant amount of wasted processing time. Now, the challenge here is how to automate these processes to ensure that all processes executed successfully.

Quality of data is also a concern as data are coming in multiple formats from different operative systems. ETL (Extract, Transform, and Load) processes do the basic pre-processing and transformation before loading into DWH. However, the quality of data is not up to the mark. Missing data or non-accurate data are also causing serious implications in many cases [4]. There are some scenarios where bad-quality data disturb the making of important decisions. Hence, the data quality is an area of concern that needs to be addressed.

3.2 Insurance Industry

Fortune 100 insurance provider that serves more than a million customers globally wants to build next-generation insurance platform [12]. Companies already launched several applications by leveraging their technical expertise that drastically changes the user experience. One such application in the car insurance industry allows the customer to pull all the repair shops in the selected radius, choose the one, get an estimate of their damage, and request a tow truck or even rental car service. The company needs the easiest way to bring its new services/features quickly to market so that they can meet the high customer expectations of continuous improvement.

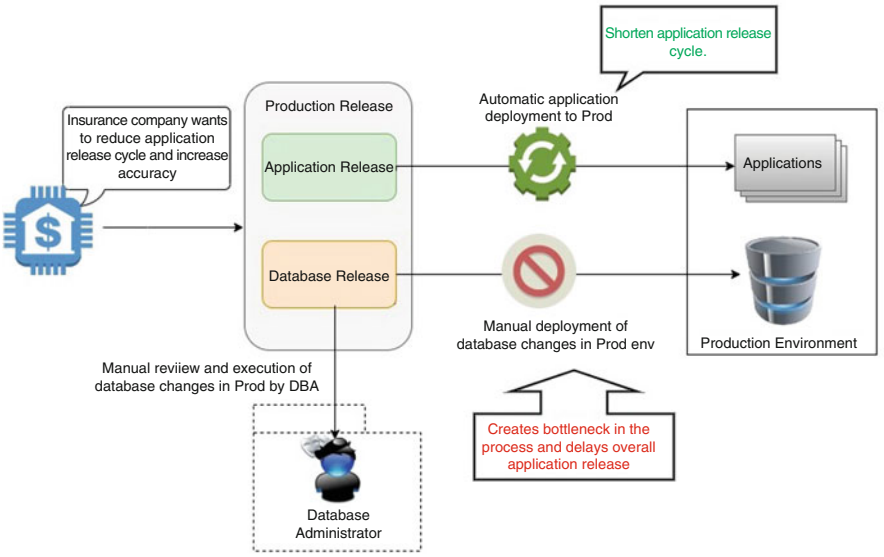


Fig. 2 Challenges in insurance industry

The insurance company has already started using advanced tools and technologies for code deployment. The overall goal of the company is to reduce the overall application release cycle. But, the manual database deployment process creates a bottleneck. This manual process takes a considerable amount of time and effort, and also deployment is risky as manual intervention is needed. In this manual process, team is facing a lot of challenges to keep track of database changes and synchronize the data across all environments as explained in Fig. 2. There is chance to miss some critical data due to the mismatch of database schema in different environments. Insurance company wants to address these issues for faster time to market with maximum accuracy. Also product team wants to receive early feedback from business.

3.3 Banking Industry

Security [13] plays an important role almost in all domains specially for banks as it handles money and personal information that are hacker's favorite. There can be a security flaw, for example, an unauthorized user accessing an application or a specific feature that s/he is not supposed to access, or a malicious user attempting to do something in order to break the system. These kinds of flaws can not be acceptable. Also, an application can behave unexpectedly due to a flaw in the logic or it can be a security problem as showcased in Fig. 3. In both cases, an instant investigation needs to be started by the security team or operation team or

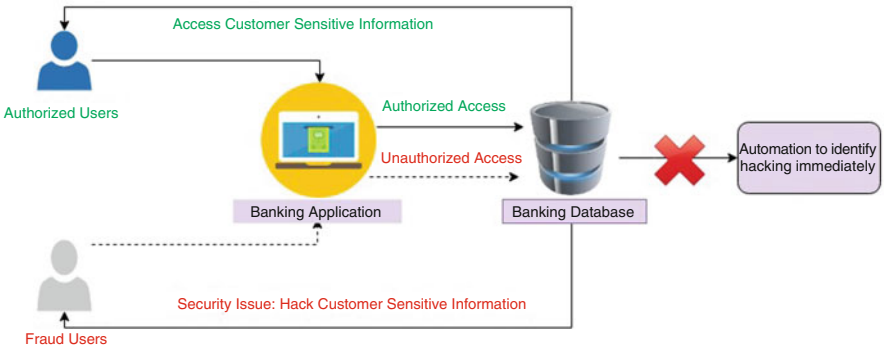


Fig. 3 Challenges in banking industry

development team in order to find out if there is a problem and how to solve it. Immediate reaction is required for security issues to prevent unauthorized access of data. If there is any code issue, patch needs to be deployed quickly in production to prevent the issue in future. The challenge is to how quickly detect the security flaws by analyzing data and how quickly take necessary action (if required, deploy code) to prevent security issue.

3.4 Aviation Industry

The mechanic crew generally checks functioning of aircraft engines [13] in flight from the ground. They examined and analyzed different data coming from different sensors on the engine to determine if aircraft engine is behaving abnormally. It is vital to take decisions on real-time data that are coming through different sensors as manifested in Fig. 4. If something abnormal happens, necessary actions need to be taken immediately to avoid any unwanted circumstances. The challenge is how to automatically update data warehouse with those data and derive information in real time from those data based on which critical operational decisions can be taken.

Problems Considered

Based on the case studies discussed above, the following problems are shown in Fig. 5, which is an important area for research, and are considered for further study and analysis.

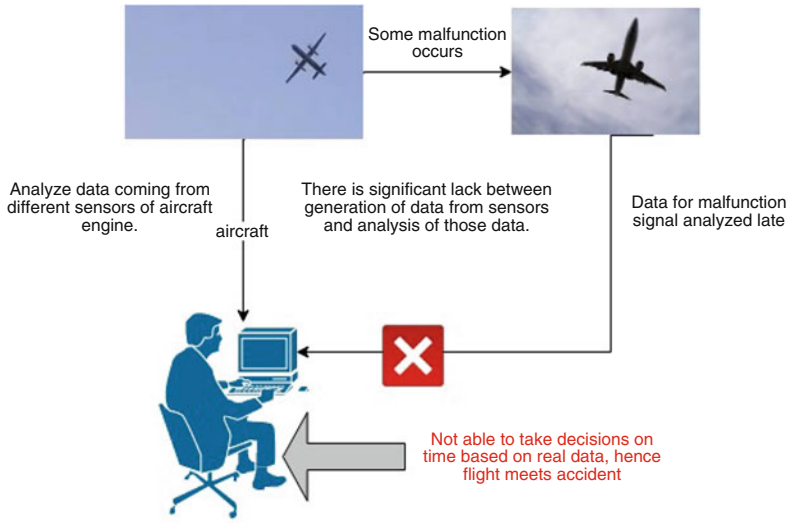


Fig. 4 Challenges in aviation industry

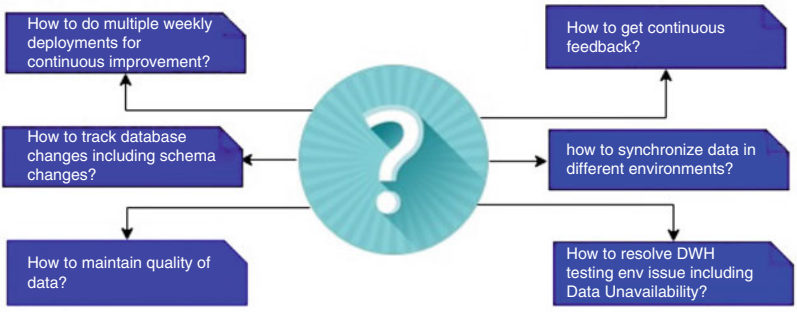


Fig. 5 Problems considered for further research and analysis

4 Proposed Solution

An architecture is designed to address the challenges faced in the practical application of legacy ETL processing. Figure 6 shows the overall architectural design of the automated ETL system. Proposed data integration steps are discussed in the next subsection. The pipeline of the proposed system is presented in Fig. 7. This pipeline explains the overall working process and development progress of the proposed solution.

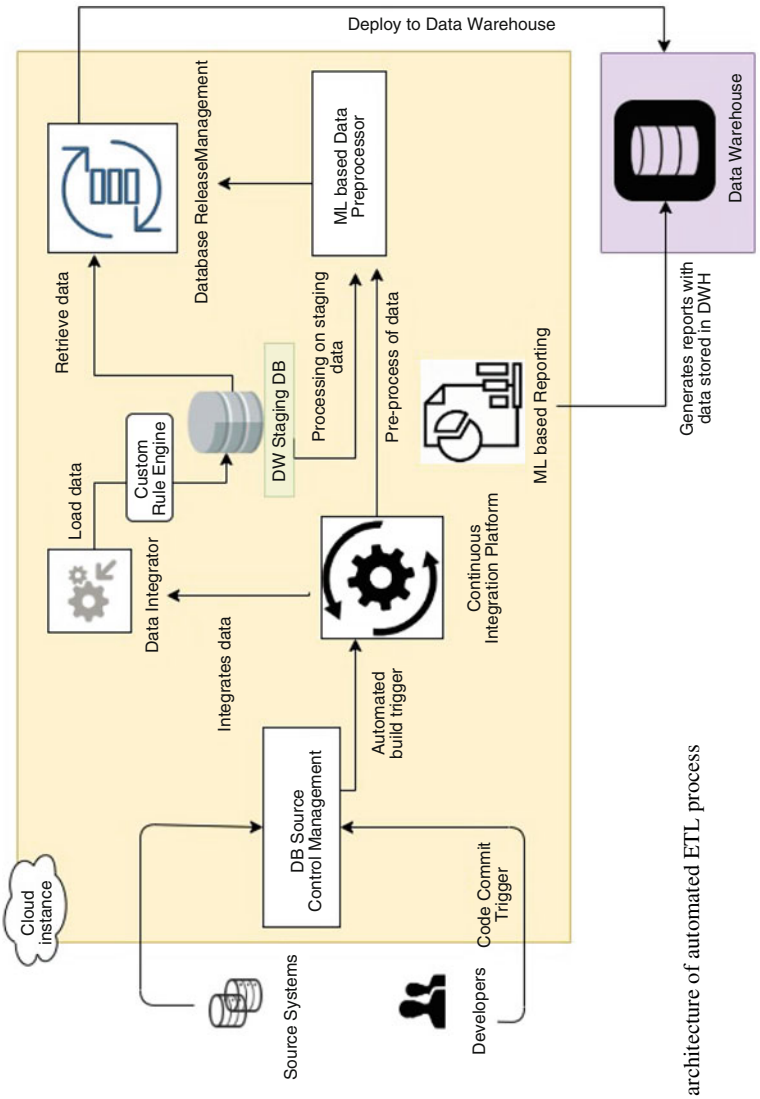


Fig. 6 Proposed architecture of automated ETL process

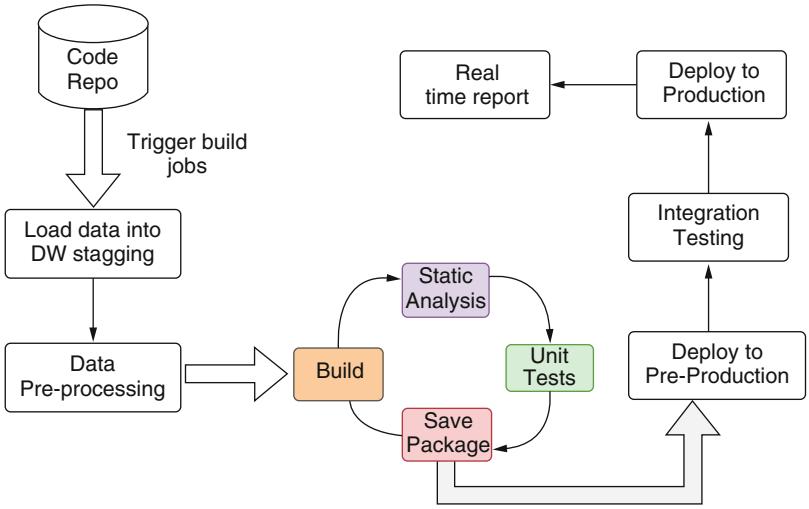


Fig. 7 Pipeline of the proposed solution process

4.1 Automated Data Integration

The main objective of this chapter is the overall automation of the data integration process. To achieve this, continuous integration (CI) [16] platform is used. Data integration tool Informatica integrates with Jenkins (https://jenkins.io/) to allow for building, testing, and releasing database changes faster and more frequently. Jenkins is an open-source CI tool that orchestrates the ETL processes with automation. Jenkins pipeline is set up to execute automated scripts to sequentially perform the following steps of the ETL process:

Trigger Build Jobs: Database scripts are considered as application code, and it should be properly version controlled. Database changes are captured and tracked in database version control. In the proposed solution, Liquibase by Datical is considered as source control for databases. Liquibase tracks the database changes including schema changes. A new changelog file is created in XML/YML/JSON/SQL format. Changeset is added in changelog file. The changelog file is committed to source control after running the Liquibase update. Jenkins hook detects this change and triggers the build process.

Load Data into DWH Staging Table: Custom rule engine would be used to classify the data as structured or unstructured using machine learning classification algorithms. Based on the type, data would be loaded in the staging database using informatica extraction and load feature.

Data Pre-Processing: Machine-learning-based custom pre-processor processes the data stored in the staging database to improve data quality.

Code Build: Instead of doing complete database build, incremental database build is done. Database code is build using Liquibase.

Automated Code Review and Analysis: Newly build code is scanned to identify coding standard violations and other issues such as code duplication, etc., through code. Proactive monitoring add-ons of Informatica review the code in an automated manner. The complex event-processing engine responds to events and does static code analysis.

Automated Test Case Generation and Unit Testing: Data validation test cases are generated using PowerCenter Data Validation Option (DVO) add-on of Informatica. Unit testing is executed in an automated manner by running pre-build test cases.

Save Package: After successful unit testing, the package is stored in a binary repository. JFrog Artifactory can be used as binary repository. Binary package can also be stored in in-build repository of Informatica.

Automated Deployment to Pre-production: Testing environment is an immutable infrastructure hosted in the cloud that is going to be created on demand for test execution. After completion of testing, the pre-prod environment is going to be deleted. Data in the pre-prod environment are also generated on demand. The executable package is retrieved from Artifactory by the Python scripts and deploys to the testing environment.

Automated Integration Testing: There are different tools available in the market to do ETL testing. Informatica Data validation (DVO) tool is used for integration testing. Test cases are generated automatically by DVO. Other tools such as QuerySurge can also be used for DWH testing.

Automated Deployment to Production: After successful integration testing, data packages are deployed in production. The production deployment happens in the same way deployment to pre-production environment happens.

Generating real-time report: Real-time reports are generated from Prod DWH using a custom machine-learning-based reporting module.

4.2 Details of Major Components

4.2.1 Database Version Control

In today's world, data are an integral part of any application since the data volume of structured and unstructured data is increasing exponentially day by day. It is going to be impossible by the traditional database systems to handle it [15]. Versioning of database is required, which maintains all the database changes. There are some tools such as Liquibase, Flyway available in the market for versioning of databases. In this chapter, we consider Liquibase as Database Source Control. It is a database-independent library that efficiently tracks and manages any database schema changes. Liquibase scripts support updating the schema of RDBMS. As

database versions are maintained, any previous version can be restored at any point in time.

4.2.2 Custom Rule Engine

Data are coming from different sources in a different format. Custom rule engine builds on machine learning can be used to specify the class to which data belong (structured or unstructured). Based on the class [1, 18] of data, appropriate rules can be used to load or transform data.

4.2.3 Data Pre-processor

Pre-processing of data is a crucial step as far as data quality is concerned. The success of the machine learning model [26] largely depends on the quality of data. This stage selects target data, prepares it by simplifying through various filtering and transformation process, and makes it available for machine-learning-based applications.

4.2.4 Database Release Automation

Database release automation (DRA) component supports seamless integration of all database changes so that the database code can be promoted along with application code changes. In today's data-driven environment, quicker database code deployment is a practical demand. With database release automation, the overall application release cycle is shortened. Enterprises can release more applications in less time, which helps them responding to customer needs quickly and adds more value to customers. Whenever the changeLog file is checked into the database source control, the automatic build is triggered in the continuous integration server. Microsoft VSTS, Bamboo, and Jenkins are some popular continuous integration tools. The database automation tool performs validation, unit tests, and creating release binaries. Test automation also needs to be added to DRA. It should be able to create immutable infrastructure on demand for testing that mirrors the production environment with required masking of data. Database release automation [20] enhances data security. Database release automation assures better database code by eliminating errors that can cause application performance issues or downtime. Common mistakes (for example, misplaced GRANT statements) that make databases more vulnerable to breaches and data theft are eliminated.

5 Solutions to Case Study Problems

Figure 8 shows a proposed solution to address the problems discussed above in the case study section in 3.

5.1 Case Study 1: Manufacturing

Challenges The electronic manufacturing company has used different IT applications for various departments such as material management, product design, production, sales and marketing, management, finance, and so on. The company is facing a lot of challenges in the current ETL process. Lots of coordination among multiple teams and manual intervention are required, which increases the risk of human-induced errors. Data quality is also not up to the mark. Hence, the objective is to streamline the ETL process and improve data quality for getting better insights from data.

Proposed Solution The automated ETL process ensures that all steps in the ETL process are executed in the correct sequence and in the correct manner. Machine-learning-based data pre-processor is used to pre-process the data more rigorously. It is reducing the processing time significantly and produces a good quality of data from the data warehouse.

5.2 Case Study 2: Insurance

Challenges Application release automation is in place for this company. But, database changes are manual and are not tracked properly. Also, manual intervention is required to handle database schema changes. So it is taking a long time to deploy changes that involve database changes. The company needs a quicker way to deploy database changes for continuous improvement.

Proposed Solution Database Version Control (Liquibase) tool is used to track database changes that can be able to identify database schema changes. This tool is integrated with the database release automation component that automates the release process of database changes. A company can support multiple weekly deployments by using the database release automation component easily.

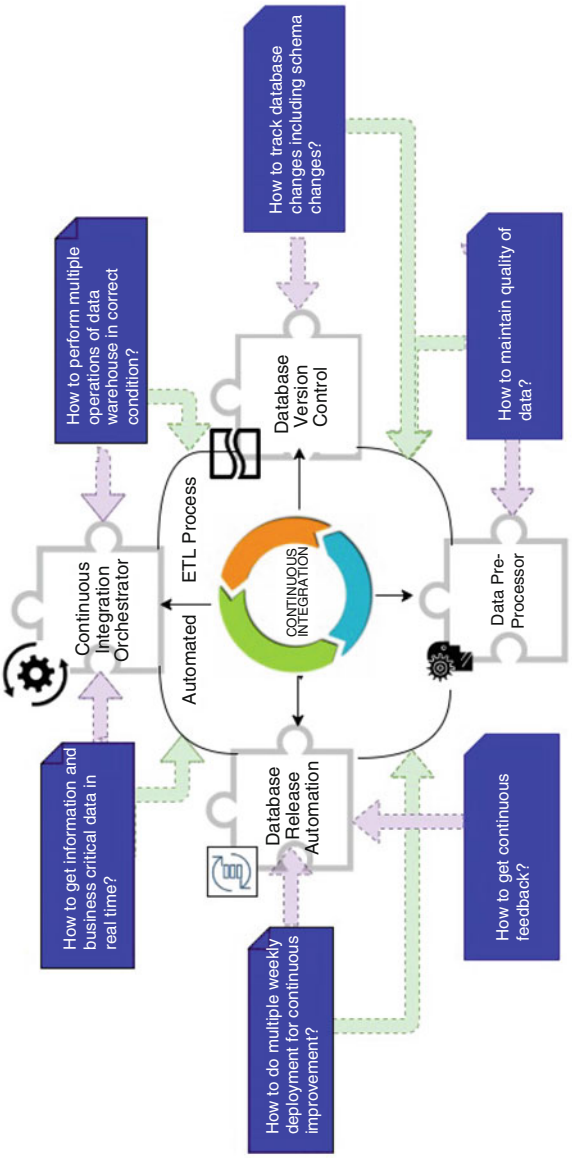


Fig. 8 Proposed solution of case study problems

5.3 Case Study 3: Banking

Challenges In case of hacking, immediate action needs to be taken to prevent attackers from accessing the bank database. Challenges are:

- How quickly can security be detected by analyzing the data.
- How quickly can necessary action be taken (if required code deployment).

Proposed Solution

- If the ETL process is automated, security flaws can be detected immediately by analyzing real-time data. Therefore, the security team/operation team/development team can start an investigation immediately to find out the issue and stop the attacker for accessing unintended data.
- If there is a code issue or database change is required, the patch needs to be applied quickly to prevent occurring this in the future. Deployment can be done quickly by application and database release automation.

5.4 Case Study 4: Aviation

Challenges Not able to analyze real-time data from different sensors of the aircraft engines. The aviation company is struggling to update data warehouses with real-time data and integrate them into BI processes. Hence, even if the aircraft engine is malfunctioning, necessary actions cannot be taken immediately to avoid accidents.

Proposed Solution Through automated ETL process, data from different sensors are loaded in the data warehouse in real time, and report has been generated based on that data. Hence, any malfunctioning of aircraft engines has been detected by the mechanic crew, and necessary actions can be taken to avoid the accident.

6 Conclusion and Future Study

We have discussed the approach to automating the ETL process so that DWH can be refreshed with updated data with minimum or no manual intervention and reports can be generated based on real-time data. The proposed approach supports automated data integration that also includes database release automation. Proof-of-concept has been performed on a continuous integration process where Liquibase is used as database source control for managing database changes. Research has also been done on the machine-learning-based automation process. Some modules including data pre-processor, custom rule engine, and reporting module have been identified where machine learning can be leveraged. As a next step, work can be carried out to integrate an ETL tool in the automation process. The study can

be continued on machine learning approaches and identification of other areas of the automated process where machine learning algorithms can be utilized to fully automate the ETL process.

References

1. Charu C. Aggarwal. *Data classification: algorithms and applications*. Chapman and Hall/CRC, 1st edition, 2014.
2. Z. El Akkaoui, E. Zimányi, J. N. Mazón López, J. C. Trujillo Mondéjar, et al. A BPMN-based design and maintenance framework for ETL processes. *International Journal of Data Warehousing and Mining (IJDWM)*, 9(3):46–72, 2013.
3. I. Ankorion. Change data capture efficient ETL for real-time BI. *DM Review*, 15(1):36, 2005.
4. Jere Aunola. Data Quality in Data Warehouses. Master's thesis, Lahti University of Applied Sciences, Last accessed January 26, 2020.
5. D P. Ballou and G K. Tayi. Enhancing data quality in data warehouse environments. *Communications of the ACM*, 42(1):73–78, 1999.
6. N. Biswas, S. Chattapadhyay, G. Mahapatra, S. Chatterjee, and K. C. Mondal. SysML based Conceptual ETL Process Modeling. In *Communications in Computer and Information Science*, pages 242–255. Springer, Singapore, 2017.
7. N. Biswas, S. Chattapadhyay, G. Mahapatra, S. Chatterjee, and K. C. Mondal. A New Approach for Conceptual ETL Process Modeling. *International Journal of Ambient Computing and Intelligence (IJACI)*, 10(1):30–45, 2019.
8. N. Biswas, A. Sarkar, and K. C. Mondal. Empirical Analysis of Programmable ETL Tools. In *Communications in Computer and Information Science*, pages 267–277. Springer, Singapore, 2018.
9. N. Biswas, A. Sarkar, and K. C. Mondal. Efficient Incremental Loading in ETL Processing for Real-Time Data Integration. *Innovations in Systems and Software Engineering*, 16:53–61, 2019.
10. M. B. Bokade, S. S. Dhande, and H. R. Vyavahare. Framework of Change Data Capture and Real Time Data Warehouse. *International Journal of Engineering Research and Technology*, 2(4), 2013.
11. M. Castellanos, A. Simitsis, K. Wilkinson, and U. Dayal. Automating the loading of business process data warehouses. In *International Conference on Extending Database Technology: Advances in Database Technology*, pages 612–623. ACM, 2009.
12. cStor. Leveraging DevOps for a Next Gen Insurance Platform. *Case Study*, Last accessed on September 07, 2020. https://cstor.com/case-study-optimize-devops-improve-customer-experience/cstor-finance-devops-case-study-thumb/.
13. Kot Dotson. The DevOps of data: How informatica prepares developers for the age of Data 3.0 #infa16. Technical report, Last accessed on September 07, 2020. https://siliconangle.com/2016/06/01/the-devops-of-data-how-informatica-prepares-developers-for-the-age-of-data-3-0-infa16/.
14. D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y. K. Ng, and R. D. Smith. Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages. *Data and Knowledge Engineering*, 31(3):227–251, 1999.
15. M. Fischer, M. Pinzger, and H. Gall. Populating a release history database from version control and bug tracking systems. In *International Conference on Software Maintenance*, pages 23–32. IEEE, 2003.
16. Informatica. Continuous Integration-Delivery-Deployment in Next Generation Data Integration. *White Paper*, Last accessed on September 07, 2020. https://kb.informatica.com/whitepapers/4/Documents.

17. W. H. Inmon. *Building the data warehouse*. John Wiley & Sons, 3rd edition, 2002.
18. S. B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: A review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering*, 160:3–24, 2007.
19. SB Kotsiantis, D. Kanellopoulos, and PE Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.
20. Liquibase. How to get started with database release automation in 4 easy steps. *White Paper*, Last accessed on September 07, 2020. https://www.datical.com/whitepapers/how-to-get-started-with-database-release-automation-4-steps/.
21. L. Muñoz, J. N. Mazón, and J. Trujillo. Automatic generation of ETL processes from conceptual models. In *International Workshop on Data Warehousing and OLAP*, pages 33–40. ACM, 2009.
22. M. A. Naeem, G. Dobbie, and G. Webber. An event-based near real-time data integration architecture. In *12th Enterprise Distributed Object Computing Conference Workshops*, pages 401–404. IEEE, 2008.
23. N. Polyzotis, S. Skiadopoulos, P. Vassiliadis, A. Simitsis, and N. Frantzell. Supporting streaming updates in an active data warehouse. In *IEEE 23rd International Conference on Data Engineering (ICDE'07)*, pages 476–485. IEEE, 2007.
24. W. Qu, V. Basavaraj, S. Shankar, and S. Dessloch. Real-Time Snapshot Maintenance with Incremental ETL Pipelines in Data Warehouses. In *Big Data Analytics and Knowledge Discovery*, pages 217–228. Springer, 2015.
25. V. Radhakrishna and K. SravanKiran, V.and Ravikiran. Automating ETL Process with Scripting Technology. In *Nirma University International Conference on Engineering (NUiCONE)*, pages 1–4. IEEE, 2012.
26. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.
27. D. Skoutas and A. Simitsis. Designing ETL Processes using Semantic Web Technologies. In *9th International Workshop on Data Warehousing and OLAP (DOLAP 2006)*, pages 67–74. ACM, 2006.
28. UC4 Software. Benefits of automating data warehousing. *White Paper*, pages 1–9, Last accessed on September 07, 2020. http://hosteddocs.ittoolbox.com/aa_data_warehouse_wp_us.pdf.
29. S. Suresh, J. P. Gautam, G. Pancha, Frank J. DeRose, and M. Sankaran. Method and architecture for automated optimization of ETL throughput in data warehousing applications, 2001. US Patent 6208990.
30. M. N. Tho and A. M. Tjoa. Zero-latency data warehousing for heterogeneous data sources and continuous data streams. In *5th International Conference on Information Integration and Web-based Applications Services*, pages 55–64, 2003.
31. V. Tziovara, P. Vassiliadis, and A. Simitsis. Deciding the physical implementation of ETL workflows. In *International Workshop on Data Warehousing and OLAP*, pages 49–56. ACM, 2007.
32. P. Vassiliadis. A Survey of Extract - Transform - Load Technology. *International Journal of Data Warehousing and Mining*, 5(3):1–27, 2009.
33. P. Vassiliadis and A. Simitsis. Near Real Time ETL. *Springer Annals of Information Systems*, 3:1–31, 2008.
34. P. Vassiliadis and A. Simitsis. Extraction, transformation, and loading. In *Encyclopedia of Database Systems*, pages 1095–1101. Springer, 2009.
35. P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. On the Logical Modeling of ETL Processes. In *International Conference on Advanced Information Systems Engineering*, pages 782–786, 2002.
36. H. Zhou, D. Yang, and Y. Xu. An ETL Strategy for Real-Time Data Warehouse. In *Practical Applications of Intelligent Systems*, pages 329–336. Springer, 2011.

Rule Induction

Jerzy W. Grzymala-Busse

1 Introduction

Rule induction is one of the most important techniques of machine learning. Since regularities hidden in data are frequently expressed in terms of rules, rule induction is one of the fundamental tools of data mining at the same time. Usually, rules are expressions of the form

$$if\ (attribute-1, value-1)\ and\ (attribute-2, value-2)\ and\ \cdots$$
$$and\ (attribute-n, value-n)\ then\ (decision, value).$$

Some rule induction systems induce more complex rules, in which values of attributes may be expressed by negation of some values or by a value subset of the attribute domain.

Data from which rules are induced are usually presented in a form similar to a table in which *cases* (or *examples*) are *labels* (or *names*) for rows and variables are labeled as *attributes* and a *decision*. We will restrict our attention to rule induction that belongs to *supervised learning*: all cases are preclassified by an expert. In different words, the decision value is assigned by an expert to each case. Attributes are independent variables and the decision is a dependent variable. A very simple example of such a table is presented as Table 1, in which attributes are: *Temperature*, *Headache*, *Weakness*, *Nausea*, and the decision is *Flu*. The set of all cases labeled by the same decision value is called a *concept*. For Table 1, case set {1, 2, 4, 5} is a concept of all cases affected by flu (for each case from this set the corresponding value of *Flu* is *yes*).

J. W. Grzymala-Busse (✉)
University of Kansas, Lawrence, KS, USA
e-mail: jerzygb@ku.edu; jerzy@ku.edu

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_4

Table 1 An example of a data set

	Attributes				Decision
Case	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	high	yes	no	yes	yes
3	normal	no	no	no	no
4	normal	yes	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	yes	no	no

Table 2 An example of an erroneous data set

	Attributes				Decision
Case	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	high	yes	no	yes	yes
3	normal	no	no	no	no
4	normal	yes	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	42.5	no	no

Table 3 An example of a data set with a numerical attribute

	Attributes				Decision
Case	Temperature	Headache	Weakness	Nausea	Flu
1	40.2	yes	yes	no	yes
2	39.8	yes	no	yes	yes
3	36.8	no	no	no	no
4	36.6	yes	yes	yes	yes
5	39.6	no	yes	no	yes
6	39.8	no	no	no	no
7	36.8	no	yes	no	no

Note that input data may be affected by errors. An example of such a data set is presented in Table 2. The case 7 has value 42.5 for weakness, an obvious error, since the attribute weakness is symbolic, with possible values yes and no. Such errors must be corrected before rule induction.

Another problem is caused by numerical attributes, for example, temperature may be represented by real numbers, as in Table 3.

Usually, numerical attributes are converted into symbolic attributes before or during rule induction. The process of converting numerical attributes into symbolic attributes is called *discretization* (or *quantization*).

Table 4 An example of a data set with missing attribute values

	Attributes				Decision
Case	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	?	yes	no	yes	yes
3	normal	no	?	no	no
4	normal	?	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	yes	no	no

Table 5 An example of an inconsistent data set

	Attributes				Decision
Case	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	high	yes	no	yes	yes
3	normal	no	no	no	no
4	normal	yes	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	yes	no	no
8	normal	no	yes	no	yes

Input data may be incomplete, i.e., some attributes may have missing attribute values, as in Table 4, where ? denotes lack of the attribute value (for example, the original value was not recorded or was erased).

Additionally, input data may be inconsistent, i.e., some cases may conflict with each other. Conflicting cases have the same attribute values yet different decision values. An example of an inconsistent data set is presented in Table 5. Cases 7 and 8 are conflicting.

In Sect. 2, a brief discussion of different rule types is presented. In the next section, a few representative rule induction algorithms are discussed. Section 4 presents the main application of rule sets, classification systems, which are used to classify new cases on the basis of induced rule sets.

2 Types of Rules

A case x is *covered* by a rule r if and only if every condition (attribute–value pair) of r is satisfied by the corresponding attribute value for x. The concept C defined by the right-hand side of rule r is *indicated* by r. We say that a concept C is *completely* covered by a rule set R if and only if for every case x from C there exists a rule r

from R such that r covers x. For a given data set, a rule set R is *complete* if and only if every concept from the data set is completely covered by R.

A rule r is *consistent* with the data set if and only if for every case x covered by r, x is a member of the concept C indicated by r. A rule set R is *consistent* with a data set if and only if every rule from R is consistent with the data set.

For example, case 1 from Table 1 is covered by the following rule r:

$$(Headache, yes) \rightarrow (Flu, yes).$$

The rule r indicates concept $\{1, 2, 4, 5\}$. Additionally, the concept $\{1, 2, 4, 5\}$ is not completely covered by the rule set consisting of r, since r covers only cases 1, 2, and 4, but the rule r is consistent with the data set from Table 1.

On the other hand, the single rule

$$(Headache, no) \rightarrow (Flu, no)$$

completely covers the concept $\{3, 6, 7\}$ in Table 1, though this rule is not consistent with the same data set. The above rule covers cases 3, 5, 6, and 7.

Any of the following two rules:

$$(Headache, yes) \,\&\, (Weakness, yes) \rightarrow (Flu, yes)$$

and

$$(Temperature, high) \,\&\, (Headache, yes) \rightarrow (Flu, yes)$$

is consistent with the data set from Table 1, but the concept $\{1, 2, 4, 5\}$ is not completely covered by the rule set consisting of the above two rules since case 5 is not covered by any rule. The first rule covers cases 1 and 4, and the second rule covers case 2.

The most frequent task of rule induction is to induce a rule set R that is complete and consistent. Such a rule set R is called *discriminant* [11]. For Table 1, the rule set consisting of the following four rules:

$$(Headache, yes) \rightarrow (Flu, yes),$$

$$(Temperature, high) \,\&\, (Weakness, yes) \rightarrow (Flu, yes),$$

$$(Temperature, normal) \,\&\, (Headache, no) \rightarrow (Flu, no),$$

$$(Headache, no) \,\&\, (Weakness, no) \rightarrow (Flu, no)$$

is discriminant.

There are many other types of rules that are used. For example, some rule induction systems induce rule sets consisting of *strong* rules, i.e., rule sets in which

every rule covers many cases. Another task is to induce *associative* rules, in which in both sides of a rule, left and right, involved variables are attributes. For Table 1, an example of such an associative rule is

$$(Nausea, yes) \rightarrow (Headache, yes).$$

3 Rule Induction Algorithms

In this section, we will assume that input data sets are free of errors, no missing attribute values are present in the input data sets, and that input data sets are consistent.

In general, rule induction algorithms may be categorized as *global* and *local*. In global rule induction algorithms, the search space is the set of all attribute values, while in local rule induction algorithms the search space is the set of attribute–value pairs.

There exist many rule induction algorithms, and we will discuss only three representative algorithms, all inducing discriminant rule sets. The first is an example of a global rule induction algorithm called LEM1 (Learning from Examples Module version 1).

3.1 LEM1 Algorithm

The algorithm LEM1, a component of the data mining system LERS (Learning from Examples using Rough Sets), is based on some rough set definitions [14, 15, 16]. Let B be a nonempty subset of the set A of all attributes. Let U denote the set of all cases. The indiscernibility relation $IND(B)$ is a relation on U defined for $x, y \in U$ by $(x, y) \in IND(B)$ if and only if for both x and y the values for all attributes from B are identical.

The indiscernibility relation $IND(B)$ is an equivalence relation. Equivalence classes of $IND(B)$ are called *elementary sets* of B. For example, for Table 1, and B = {Temperature, Headache}, elementary sets of $IND(B)$ are {1}, {2}, {3, 7}, {4}, {5, 6}.

The family of all B-elementary sets will be denoted B^*, for example, in Table 1,

$$\{Temperature, Headache\}^* = \{\{1\}, \{2\}, \{3, 7\}, \{4\}, \{5, 6\}\}.$$

For a decision d, we say that $\{d\}$ depends on B if and only if $B^* \leq \{d\}^*$. A *global covering* (or *relative reduct*) of $\{d\}$ is a subset B of A such that $\{d\}$ depends on B and B is minimal in A. Thus, global coverings of $\{d\}$ are computed by comparing partitions B^* with $\{d\}^*$. The algorithm to compute a single global covering is presented below.

Algorithm to compute a single global covering
(**input**: the set $A$of all attributes, partition $\{d\}^*$ on U;
output: a single global covering R);
begin
compute partition A^*;
$P := A$;
$R := \emptyset$;
if $A^* \leq \{d\}^*$
then
begin
for each attribute a in A **do**
begin
$Q := P - \{a\}$;
compute partition Q^*;
if $Q^* \leq \{d\}^*$ **then** $P := Q$
end {for}
$R := P$
end {then}
end {algorithm}.

On the basis of a global covering, rules are computed using the *dropping conditions* technique [11]. For a rule of the form

$$C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_n \;\rightarrow\; D$$

dropping conditions means scanning the list of all conditions, from the left to the right, with an attempt to drop any condition, checking against the decision table where the simplified rule does not violate consistency of the discriminant description.

For Table 1,

$$\{Temperature,\ Headache,\ Weakness,\ Nausea\}^* =$$

$$\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}\},$$

$$\{Flu\}^* = \{\{1, 2, 4, 5\}, \{3, 6, 7\}\},$$

and

$$\{Temperature,\ Headache,\ Weakness,\ Nausea\}^* \leq \{Flu\}^*.$$

Next we need to check whether

$$\{Headache,\ Weakness,\ Nausea\}^* \leq \{Flu\}^*.$$

This condition is false since

$$\{Headache,\ Weakness,\ Nausea\}^* =$$
$$\{\{1\}, \{2\}, \{3, 6\}, \{4\}, \{5, 7\}\}.$$

Then we compute

$$\{Temperature,\ Weakness,\ Nausea\}^* =$$
$$\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}\}.$$

We observe that

$$\{Temperature,\ Weakness,\ Nausea\}^* \leq \{Flu\}^*.$$

The next partition to compute is

$$\{Temperature,\ Nausea\}^*,$$

equal to

$$\{\{1\}, \{2\}, \{3, 7\}, \{4\}, \{5, 6\}\},$$

and

$$\{Temperature,\ Nausea\}^* \not\leq \{Flu\}^*.$$

The last step is to compute

$$\{Temperature,\ Weakness\}^*,$$

equal to

$$\{\{1\}, \{2, 6\}, \{3\}, \{4, 7\}, \{5\}\},$$

and since

$$\{Temperature,\ Weakness\}^* \not\leq \{Flu\}^*,$$

the total covering is

$$\{Temperature,\ Weakness,\ Nausea\}.$$

The first case from Table 1 implies the following preliminary rule:

$$(Temperature, very\_high)\ \&\ (Weakness, yes)\ \&\ (Nausea, no) \rightarrow (Flu, yes).$$

The above rule covers only the first case. The first condition,

$$(Temperature, very\_high),$$

cannot be dropped since the rule

$$(Weakness, yes)\ \&\ (Nausea, no) \rightarrow (Flu, yes)$$

covers cases 1 and 7 from different concepts. However, an attempt to drop the next condition, $(Weakness, yes)$, is successful since the rule

$$(Temperature, very\_high)\ \&\ (Nausea, no) \rightarrow (Flu, yes)$$

covers only case 1. The next possibility, to drop the last condition $(Weakness, yes)$, is successful as well, since the resulting rule

$$(Temperature, very\_high) \rightarrow (Flu, yes)$$

covers only case 1.

In a similar way, the remaining rules are induced. The final rule set, induced by LEM1, is

$(Temperature, very\_high) \rightarrow (Flu, yes)$,
$(Nausea, yes) \rightarrow (Flu, yes)$,
$(Temperature, high)\ \&\ (Weakness, yes) \rightarrow (Flu, yes)$,
$(Weakness, no)\ \&\ (Nausea, no) \rightarrow (Flu, no)$,
$(Temperature, normal)\ \&\ (Nausea, no) \rightarrow (Flu, no)$.

For Table 1, the second global covering is

$$\{Temperature,\ Headache,\ Weakness\}.$$

3.2 LEM2 Algorithm

An idea of blocks of attribute–value pairs is used in the rule induction algorithm LEM2 (Learning from Examples Module, version 2), another component of LERS. The option LEM2 of LERS is most frequently used since—in most cases—it gives better results. LEM2 explores the search space of attribute–value pairs. Its input data file is a lower or upper approximation of a concept (for definitions of lower

and upper approximations of a concept, see, e.g., [5]), so its input data file is always consistent. In general, LEM2 computes a local covering and then converts it into a rule set. We will quote a few definitions to describe the LEM2 algorithm [2, 4].

For an attribute–value pair $(a,\ v) = t$, a *block* of t, denoted by $[t]$, is a set of all cases from U such that attribute a has value v. Let B be a nonempty lower or upper approximation of a concept represented by a decision–value pair (d, w). Set B *depends* on a set T of attribute–value pairs $t = (a, v)$ if and only if

$$\emptyset \neq [T] = \bigcap_{t \in T} [t] \subseteq B.$$

Set T is a *minimal complex* of B if and only if B depends on T, and no proper subset T' of T exists such that B depends on T'. Let $\mathcal{T}$ be a nonempty collection of nonempty sets of attribute–value pairs. Then $\mathcal{T}$ is a *local covering* of B if and only if the following conditions are satisfied:

(1) Each member T of $\mathcal{T}$ is a minimal complex of B.
(2) $\bigcup_{t \in \mathcal{T}} [T] = B$, and
$\mathcal{T}$ is minimal, i.e., $\mathcal{T}$ has the smallest possible number of members.

The procedure LEM2 is presented below.

Procedure LEM2
(**input**: a set B,
output: a single local covering $\mathcal{T}$ of set B);
begin
$G := B$;
$\mathcal{T} := \emptyset$;
while $G \neq \emptyset$
begin
$T := \emptyset$;
$T(G) := \{t | [t] \cap G \neq \emptyset\}$;
while $T = \emptyset$ **or** $[T] \not\subseteq B$
begin
select a pair $t \in T(G)$ such that $|[t] \cap G|$ is
maximum; if a tie occurs, select a pair $t \in T(G)$
with the smallest cardinality of $[t]$;
if another tie occurs, select first pair;
$T := T \cup \{t\}$;
$G := [t] \cap G$;
$T(G) := \{t | [t] \cap G \neq \emptyset\}$;
$T(G) := T(G) - T$;
end {while}
for each $t \in T$ **do**
if $[T - \{t\}] \subseteq$ B **then** $T := T - \{t\}$;
$\mathcal{T} := \mathcal{T} \cup \{T\}$;

$G := B - \cup_{T \in \mathcal{T}}[T]$;
end {while};
for each $T \in \mathcal{T}$ **do**
if $\bigcup_{S \in \mathcal{T}-\{T\}}[S] = B$ **then** $\mathcal{T} := \mathcal{T} - \{T\}$;
end {procedure}.

For a set X, $|X|$ denotes the cardinality of X.
The first step of the algorithm LEM2 is to compute all attribute–value pair blocks. For Table 1, these blocks are
$[(Temperature, very\_high)] = \{1\}$,
$[(Temperature, high)] = \{2, 5, 6\}$,
$[(Temperature, normal)] = \{3, 4, 7\}$,
$[(Headache, yes)] = \{1, 2, 4\}$,
$[(Headache, no)] = \{3, 5, 6, 7\}$,
$[(Weakness, yes)] = \{1, 4, 5, 7\}$,
$[(Weakness, no)] = \{2, 3, 6\}$,
$[(Nausea, no)] = \{1, 3, 5, 6, 7\}$,
$[(Nausea, yes)] = \{2, 4\}$.
Let us induce rules for the concept $\{1, 2, 4, 5\}$. Hence, $B = G = \{1, 2, 4, 5\}$. The set $T(G)$ of all relevant attribute–value pairs is

$$\{(Temperature, very\_high), (Temperature, high),$$
$$(Temperature, normal), (Headache, yes),$$
$$(Headache, no), (Weakness, yes),$$
$$(Weakness, no), (Nausea, no), (Nausea, yes)\}.$$

The next step is to identify attribute–value pairs (a, v) with the largest $|[(a, v)] \cap G|$. For two attribute–value pairs from $T(G)$, $(Headache, yes)$ and (*Weakness, yes*), the cardinality of the set $|[(a, v)] \cap G|$ is equal to three. The next criterion is the size of the attribute–value pair block, this size is smaller for $(Headache, yes)$ than for $(Weakness, yes)$, so we select $(Headache, yes)$. Besides, $[(Headache, yes)] \subseteq B$, so $(Headache, high)$ is the first minimal complex of G.
The new set G is equal to $B[(Headache, yes)] = \{1, 2, 4, 5\} - \{1, 2, 4\} = \{5\}$. A new set $T(G)$ is equal to

$$\{(Temperature, high), (Headache, no), (Weakness, yes), (Nausea, no)\}.$$

This time the first criterion, the largest $|[(a, v)] \cap G|$, identifies all four attribute–value pairs. The second criterion, the size of the attribute–value block, selects $(Temperature, high)$. However,

$$[(Temperature,\ high)] = \{2, 5, 6\} \not\subseteq B,$$

so we have to go through an additional iteration of the internal loop. The next candidates are $(Headache,\ no)$ and $(Weakness,\ yes)$, since for both of these attribute–value pairs the sizes of their blocks are equal to four. On the basis of heuristics, we will select $(Headache,\ no)$. But

$$[(Temperature,\ high)] \cap [(Headache, no)] = \{5, 6\} \not\subseteq B = \{1, 2, 4, 5\},$$

so we have to add $(Weakness, yes)]$ as well. This time

$$[(Temperature,\ high)] \cap [(Headache, no)] \cap [(Weakness, yes)] =$$
$$\{5\} \subseteq B = \{1, 2, 4, 5\},$$

so our candidate for a minimal complex is the set

$$\{(Temperature,\ high), (Headache,\ no), (Weakness, yes)\}.$$

We have to run the following part of the LEM2 algorithm:

for each $t \in T$ **do**
  if $[T - \{t\}] \subseteq$ B **then** $T := T - \{t\}$;

As a result, the second minimal complex is identified:

$$\{(Temperature,\ high), (Weakness, yes)\}.$$

Eventually, the local covering of $B = \{1, 2, 4, 5\}$ is the set

$$\{\{(Headache,\ yes)\}, \{(Temperature,\ high), (Weakness, yes)\}\}.$$

The complete rule set, induced by LEM2, is
$(Headache, yes) \rightarrow (Flu, yes)$,
$(Temperature, high)$ & $(Weakness, yes) \rightarrow (Flu, yes)$,
$(Temperature, normal)$ & $(Headache, no) \rightarrow (Flu, no)$,
$(Headache, no)$ & $(Weakness, no) \rightarrow (Flu, no)$.

Obviously, in general, rule sets induced by LEM1 differ from rule sets induced by LEM2 from the same data sets. Additionally, we may trace rule induction by LEM2 for the concept {1, 2, 4, 5} using Tables 6 and 7. The corresponding comments are:

1. The set G = {1, 2, 4, 5}. The best attribute–value pairs t, i.e., with the largest cardinality of the intersection of $[t]$ and G (presented in the third column of Table 6), are (*Headache*, *yes*) and (*Weakness*, *yes*). The size of the attribute–value block is smaller for (*Headache*, *yes*), so this pair is selected (bulleted). Also, [(*Headache*, *yes*)] = {1, 2, 4} $\subseteq$ {1, 2, 4, 5} = B; hence, this rule is complete.

Table 6 Computing a local covering for the concept [(*Flu*, *yes*)], part I

$(a, v) = t$	$[(a, v)]$	{1, 2, 4, 5}
(*Temperature*, *very_high*)	{1}	{1}
(*Temperature*, *high*)	{2, 5, 6}	{2, 5}
(*Temperature*, *normal*)	{3, 4, 7}	{4}
(*Headache*, *yes*)	{1, 2, 4}	{1, 2, 4}•
(*Headache*, *no*)	{3, 5, 6, 7}	{5}
(*Weakness*, *yes*)	{1, 4, 5, 7}	{1, 4, 5}
(*Weakness*, *no*)	{2, 3, 6}	{2}
(*Nausea*, *no*)	{1, 3, 5, 6, 7}	{1, 5}
(*Nausea*, *yes*)	{2, 4}	{2, 4}
Comments		1

Table 7 Computing a local covering for the concept [(*Flu*, *yes*)], part II

$(a, v) = t$	$[(a, v)]$	{5}	{5}	{5}
(*Temperature*, *very_high*)	{1}	–	–	–
(*Temperature*, *high*)	{2, 5, 6}	{5}•	–	–
(*Temperature*, *normal*)	{3, 4, 7}	–	–	–
(*Headache*, *yes*)	{1, 2, 4}	–	–	–
(*Headache*, *no*)	{3, 5, 6, 7}	{5}	{5}•	–
(*Weakness*, *yes*)	{1, 4, 5, 7}	{5}	{5}	{5}•
(*Weakness*, *no*)	{2, 3, 6}	–	–	–
(*Nausea*, *no*)	{1, 3, 5, 6, 7}	{5}	{5}	{5}
(*Nausea*, *yes*)	{2, 4}	–	–	–
Comments		2	3	4

2. The new set $G = B - [T] = \{1, 2, 4, 5\} - \{1, 2, 4\} = \{5\}$. There are four relevant attribute–value pairs. For (*Temperature*, *high*), cardinality of $[t]$ is the smallest (and equal to 3), so we select this attribute–value pair. This time $\{2, 5, 6\} \not\subseteq \{1, 2, 4, 5\}$, so we need another iteration of the LEM2 algorithm.
3. There is a tie between cardinalities of blocks of attribute–value pairs for (*Headache*, *no*) and (*Weakness*, *yes*). We select heuristically the top pair (*Headache*, *no*). However, [(*Temperature*, *high*)] $\cap$ [(*Headache*, *no*)] = $\{5, 6\} \not\subseteq \{1, 2, 4, 5\}$; hence, we need to look for the next t.
4. The pair (*Weakness*, *yes*) is the best choice, and [(*Temperature*, *high*)] $\cap$ [(*Headache*, *no*)] $\cap$ [(*Weakness*, *yes*)] = {5}, so we identified three candidates for the second rule: (*Temperature*, *high*), (*Headache*, *no*), and (*Weakness*, *yes*). It is obvious that (*Temperature*, *high*) is a redundant condition.

3.3 MLEM2 Algorithm

The MLEM2 algorithm (Modified LEM2) may be used for rule induction from data sets with numerical attributes and missing attribute values [6]. We will present the MLEM2 algorithm that may be applied to numerical attributes. Table 3 presents an example of such a data set. First, for every numerical attribute, its domain is sorted. The cutpoints are selected as means of any two consecutive values of the sorted attribute. For each cutpoint c, two blocks are created, the first block contains all cases where values of the numerical attribute are smaller than c, and the second block contains remaining cases. Once such blocks are computed, rule induction in MLEM2 is conducted the same way as in LEM2. Tables 8 and 9 present computation of the local covering for the concept [(*Flu*, *yes*)]. The corresponding rules are

$(Headache, yes) \rightarrow (Flu, yes)$,

$(Temperature, 38.2..40.2)$ & $(Temperature, 36.6..39.7) \rightarrow (Flu, yes)$.

The second rule may be presented as follows:

$(Temperature, 38.2..39.7) \rightarrow (Flu, yes)$.

Time complexity of the rule induction algorithms LEM1, LEM2, and MLEM2, in the worst case is $O(m^2n^2)$, where m is the number of cases and n is the number of attributes [7].

3.4 AQ Algorithm

Another rule induction algorithm, developed by R. S. Michalski and his collaborators in the early seventies, is an algorithm called AQ. Many versions of the algorithm have been developed, under different names [12, 13].

Table 8 Computing a local covering for the concept [(*Flu*, *yes*)], part I

$(a, v) = t$	$[(a, v)]$	{1, 2, 4, 5}
(*Temperature*, 36.6..36.7)	{4}	{4}
(*Temperature*, 36.7..40.2)	{1, 2, 3, 5, 6, 7}	{1, 2, 5}
(*Temperature*, 36.6..38.2)	{3, 4, 7}	{4}
(*Temperature*, 38.2..40.2)	{1, 2, 5, 6}	{1, 2, 5}
(*Temperature*, 36.6..39.7)	{3, 4, 5, 7}	{4, 5}
(*Temperature*, 39.7..40.2)	{1, 2, 6}	{1, 2}
(*Temperature*, 36.6..40)	{2, 3, 4, 5, 6, 7}	{2, 4, 5}
(*Temperature*, 40..40.2)	{1}	{1}
(*Headache*, *yes*)	{1, 2, 4}	{1, 2, 4}•
(*Headache*, *no*)	{3, 5, 6, 7}	{5}
(*Weakness*, *yes*)	{1, 4, 5, 7}	{1, 4, 5}
(*Weakness*, *no*)	{2, 3, 6}	{2}
(*Nausea*, *no*)	{1, 3, 5, 6, 7}	{1, 5}
(*Nausea*, *yes*)	{2, 4}	{2, 4}

Table 9 Computing a local covering for the concept [(*Flu*, *yes*)], part II

$(a, v) = t$	$[(a, v)]$	{5}	{5}
(*Temperature*, 36.6..36.7)	{4}	–	–
(*Temperature*, 36.7..40.2)	{1, 2, 3, 5, 6, 7}	{5}	–
(*Temperature*, 36.6..38.2)	{3, 4, 7}	–	–
(*Temperature*, 38.2..40.2)	{1, 2, 5, 6}	{5}•	–
(*Temperature*, 36.6..39.7)	{3, 4, 5, 7}	{5}	{5}•
(*Temperature*, 39.7..40.2)	{1, 2, 6}	–	–
(*Temperature*, 36.6..40)	{2, 3, 4, 5, 6, 7}	{5}	{5}
(*Temperature*, 40..40.2)	{1}	–	–
(*Headache*, *yes*)	{1, 2, 4}	–	–
(*Headache*, *no*)	{3, 5, 6, 7}	{5}	{5}
(*Weakness*, *yes*)	{1, 4, 5, 7}	{5}	{5}
(*Weakness*, *no*)	{2, 3, 6}	–	–
(*Nausea*, *no*)	{1, 3, 5, 6, 7}	{5}	{5}
(*Nausea*, *yes*)	{2, 4}	–	–

Let us start by quoting some definitions from [12, 13]. Let A be the set of all attributes,
$A = \{A_1, A_2, \ldots, A_k\}$. A *seed* is a member of the concept, i.e., a positive case. A *selector* is an expression that associates a variable (attribute or decision) to a value of the variable, e.g., a negation of value, a disjunction of values, etc. A *complex* is a conjunction of selectors. A *partial star* $G(e|e_1)$ is a set of all complexes describing the seed $e = (x_1, x_2, \ldots, x_k)$ and not describing a negative case $e_1 = (y_1, y_2, \ldots, y_k)$. Thus, the complexes of $G(e|e1)$ are conjunctions of selectors of the form $(A_i, \neg y_i)$, for all i such that $x_i \neq y_i$. A *star* $G(e|F)$ is constructed from all partial stars $G(e|e_i)$, for all $e_i \in F$, and by conjuncting these partial stars by each other, using absorption law to eliminate redundancy. For a given concept C, a *cover* is a disjunction of complexes describing all positive cases from C and not describing any negative cases from $F = U - C$.

The main idea of the AQ algorithm is to generate a cover for each concept by computing stars and selecting from them single complexes to the cover.

For the example from Table 1, and concept $C = \{1, 2, 4, 5\}$ described by (Flu, yes), set F of negative cases is equal to 3, 6, 7. A seed is any member of C, say that it is case 1. Then the partial star $G(1|3)$ is equal to

$$\{(Temperature, \neg normal), (Headache, \neg no), (Weakness, \neg no)\}.$$

Obviously, partial star $G(1|3)$ describes negative cases 6 and 7. The partial star $G(1|6)$ equals

$$\{(Temperature, \neg high), (Headache, \neg no), (Weakness, \neg no)\}$$

The conjunct of $G(1|3)$ and $G(1|6)$ is equal to

$$\{(Temperature, very\_high),$$
$$(Temperature, \neg normal) \ \& \ (Headache, \neg no),$$
$$(Temperature, \neg normal) \ \& \ (Weakness, \neg no),$$
$$(Temperature, \neg high) \ \& \ (Headache, \neg no),$$
$$(Headache, \neg no),$$
$$(Headache, \neg no) \ \& \ (Weakness, \neg no),$$
$$(Temperature, \neg high) \ \& \ (Weakness, \neg no),$$
$$(Headache, \neg no) \ \& \ Weakness, \neg no),$$
$$(Weakness, \neg no)\},$$

and after using the absorption law, this set is reduced to the following set $G(1|\{3, 6\})$:

$$\{(Temperature, very\_high), (Headache \neg no), (Weakness, \neg no)\}.$$

The preceding set describes negative case 7. The partial star $G(1|7)$ is equal to

$$\{(Temperature, \neg normal), Headache, \neg no)\}.$$

The conjunct of $G(1|\{3, 6\})$ and $G(1|7)$ is

$$\{(Temperature, very\_high),$$
$$(Temperature, very\_high) \ \& \ (Headache, \neg no),$$
$$(Temperature, \neg normal) \ \& \ Headache, \neg no),$$
$$(Headache, \neg no),$$
$$(Temperature, \neg normal) \ \& \ (Weakness, \neg no),$$
$$(Headache, \neg no) \ \& \ (Weakness, \neg no)\}.$$

The above set, after using the absorption law, is already a star $G(1|F)$

$$\{(Temperature, very\_high),$$
$$(Headache, \neg no),$$
$$(Temperature, \neg normal) \ \& \ (Weakness, \neg no)\}.$$

The first complex describes only one positive case 1, while the second complex describes three positive cases: 1, 2, and 4. The third complex describes two positive cases: 1 and 5. Therefore, the complex

$$(Headache, \neg no)$$

should be selected to be a member of the star of C. The corresponding rule is

$$(Headache, \neg no) \rightarrow (Flu, yes).$$

If rules without negation are preferred, the preceding rule may be replaced by the following rule:

$$(Headache, yes) \rightarrow (Flu, yes).$$

The next seed is case 5, and the partial star $G(5|3)$ is the following set

$$\{(Temperature, \neg normal), (Weakness, \neg no)\}.$$

The partial star $G(5|3)$ covers cases 6 and 7. Therefore, we compute $G(5|6)$, equal to

$$\{(Weakness, \neg no)\}.$$

A conjunct of $G(5|3)$ and $G(5|6)$ is the following set:

$$\{(Temperature, \neg normal) \& (Weakness, \neg no), (Weakness, \neg no)\}.$$

After simplification, the set $G(5|\{3, 6\})$ equals

$$\{Weakness, \neg no)\}.$$

The above set covers case 7. The set $G(5|7)$ is equal to

$$\{(Temperature, \neg normal)\}.$$

Finally, the partial star $G(5|\{3, 6, 7\})$ is equal to

$$\{(Temperature, \neg normal) \& (Weakness, \neg no)\},$$

so the second rule describing concept $\{1, 2, 4, 5\}$ is

$$(Temperature, \neg normal) \& (Weakness, \neg no) \rightarrow (Flu, yes).$$

It is not difficult to see that the following rules describe the second concept from Table 1:

$$(Temperature, \neg high)\ \&\ (Headache, \neg yes) \rightarrow (Flu, no),$$

$$(Headache, \neg yes)\ \&\ (Weakness, \neg yes) \rightarrow (Flu, no).$$

Note that the AQ algorithm demands computing conjuncts of partial stars. In the worst case, time complexity of this computation is $O(n^m)$, where n is the number of attributes and m is the number of cases. The authors of AQ suggest using the parameter MAXSTAR as a method of reducing the computational complexity. According to this suggestion, any set, computed by conjunction of partial stars, is reduced in size if the number of its members is greater than MAXSTAR. Obviously, the quality of the output of the algorithm is reduced as well.

4 Classification Systems

Rule sets, induced from data sets, are used mostly to classify new, unseen cases. Such rule sets may be used in rule-based expert systems.

There is a few existing classification systems, e.g., associated with rule induction systems LERS or AQ. A classification system used in LERS is a modification of the well-known bucket brigade algorithm [1, 8, 18]. In the rule induction system AQ, the classification system is based on a rule estimate of probability [13, 12]. Some classification systems use a decision list, in which rules are ordered, and the first rule that matches the case classifies it [17]. In this section, we will concentrate on a classification system associated with LERS.

The decision to which concept a case belongs is made on the basis of three factors: *strength*, *specificity*, and *support*. These factors are defined as follows: *strength* is the total number of cases correctly classified by the rule during training. *Specificity* is the total number of attribute–value pairs on the left-hand side of the rule. The matching rules with a larger number of attribute–value pairs are considered more specific. The third factor, *support*, is defined as the sum of products of strength and specificity for all matching rules indicating the same concept. The concept C for which the support, i.e., the following expression

$$\sum_{matching\ rules\ r\ describing\ C} Strength(r) * Specificity(r)$$

is the largest is the winner and the case is classified as being a member of C.

In the classification system of LERS, if complete matching is impossible, all partially matching rules are identified. These are rules with at least one attribute–value pair matching the corresponding attribute–value pair of a case. For any partially matching rule r, the additional factor, called *Matching_factor* (r), is computed. Matching_factor (r) is defined as the ratio of the number of matched attribute–value

pairs of r with a case to the total number of attribute–value pairs of r. In partial matching, the concept C for which the following expression is the largest

$$\sum_{\substack{\text{partially matching} \\ \text{rules } r \text{ describing } C}} Matching\_factor(r) * Strength(r) * Specificity(r)$$

is the winner and the case is classified as being a member of C.

5 Validation

The most important performance criterion of rule induction methods is the error rate. A complete discussion on how to evaluate the error rate from a data set is contained in [19]. If the number of cases is less than 100, the *leaving-one-out* method is used to estimate the error rate of the rule set. In leaving-one-out, the number of learn-and-test experiments is equal to the number of cases in the data set. During the i-th experiment, the i-th case is removed from the data set, a rule set is induced by the rule induction system from the remaining cases, and the classification of the omitted case by rules produced is recorded. The error rate is computed as

$$\frac{total\ number\ of\ misclassifications}{number\ of\ cases}.$$

On the other hand, if the number of cases in the data set is greater than or equal to 100, the *ten-fold cross-validation* will be used. This technique is similar to leaving-one-out in that it follows the learn-and-test paradigm. In this case, however, all cases are randomly re-ordered, and then a set of all cases is divided into ten mutually disjoint subsets of approximately equal size. For each subset, all remaining cases are used for training, i.e., for rule induction, while the subset is used for testing. This method is used primarily to save time at the negligible expense of accuracy.

Ten-fold cross-validation is commonly accepted as a standard way of validating rule sets. However, using this method twice, with different preliminary random re-ordering of all cases, yields—in general—two different estimates for the error rate [5].

For large data sets (at least 1000 cases), a single application of the train-and-test paradigm may be used. This technique is also known as *holdout* [19]. Two thirds of cases should be used for training and one third for testing.

6 Advanced Methodology

Some more advanced methods of machine learning in general and rule induction in particular were discussed in [3]. Such methods include combining a few rule sets with associated classification systems, created independently, using different algorithms, to classify a new case by taking into account all individual decisions and using some mechanisms to resolve conflicts, e.g., voting [10]. Another important problem is scaling up rule induction algorithms. Yet another important problem is learning from imbalanced data sets [9], where some concepts are extremely small.

References

1. Booker, L.B., Goldberg, D.E., F., H.J.: Classifier systems and genetic algorithms. MIT Press, Boston (1990)
2. Chan, C.C., Grzymala-Busse, J.W.: On the attribute redundancy and the learning programs ID3, PRISM, and LEM2. Tech. rep., Department of Computer Science, University of Kansas (1991)
3. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Machine Learning **40**, 139–157 (2000)
4. Grzymala-Busse, J.W.: LERS—a system for learning from examples based on rough sets. In: Slowinski, R. (ed.) Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory, pp. 3–18. Kluwer Academic Publishers, Dordrecht, Boston, London (1992)
5. Grzymala-Busse, J.W.: A new version of the rule induction system LERS. Fundamenta Informaticae **31**, 27–39 (1997)
6. Grzymala-Busse, J.W.: MLEM2: A new algorithm for rule induction from imperfect data. In: Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. pp. 243–250 (2002)
7. Grzymala-Busse, J.W., Clark, P.G., Kuehnhausen, M.: Generalized probabilistic approximations of incomplete data. International Journal of Approximate Reasoning **132**, 180–196 (2014)
8. Holland, J.H., Holyoak, K.J., Nisbett, R.E.: Induction. Processes of Inference, Learning, and Discovery. MIT Press, Boston (1986)
9. Japkowicz, N.: Learning from imbalanced data sets: A comparison of various strategies. In: Learning from Imbalanced Data Sets, AAAI Workshop at the 17th Conference on AI, AAAI-2000. pp. 10–17 (2000)
10. Kuncheva, L.I.: Combining Pattern Classifiers. Methods and Algorithms. John Wiley & Sons, Hoboken, NJ (2004)
11. Michalski, R.S.: A theory and methodology of inductive learning. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.) Machine Learning and Artificial Intelligence Approach, pp. 83–134. Morgan Kaufmann, San Mateo, CA (1983)
12. Michalski, R.S., Mozetic, I., Hong, J., Lavrac, N.: The multi-purpose incremental learning system AQ15 and its testing application on three medical domains. In: Proceedings of the National Conference on Artificial Intelligence. pp. 1041–1045. Morgan Kaufmann, San Mateo, CA (1986)
13. Michalski, R.S., Mozetic, I., Hong, J., N., L.: The AQ inductive learning system; an overview and experiments. Tech. rep., Department of Computer Science, University of Illinois at Urbana-Champaign (1986)
14. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences **11**, 341–356 (1982)

15. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht, Boston, London (1991)
16. Pawlak, Z., Grzymala-Busse, J.W., Slowinski, R., Ziarko, W.: Rough sets. Communications of the ACM **38**, 89–95 (1995)
17. Rivest, R.L.: Learning decision lists. Machine learning **2**, 229–246 (1987)
18. Stefanowski, J.: Algorithms of Decision Rule Induction in Data Mining. Poznan University of Technology Press, Poznan, Poland (2001)
19. Weiss, S., Kulikowski, C.A.: Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems. Morgan Kaufmann Publ., San Mateo, CA (1991)

Nearest-Neighbor Methods: A Modern Perspective

Aryeh Kontorovich and Samory Kpotufe

1 Introduction

Given the ancient origins of nearest-neighbor-based prediction,[1] it is perhaps surprising that this seemingly naive approach remains competitive in some cases against the state-of-the-art techniques [9, 12, 48, 62]. Even a cursory historical account of NN methods would be outside of the scope of this chapter, and we recommend that the reader consults the excellent recent monographs [8, 12] as well as more classic texts [22, 35] for an exhaustive survey. Rather, our scope narrowly will focus on providing the data science practitioner with an overview of the most recent NN-based algorithms, as well as brief synopses of their runtimes, underlying assumptions and convergence guarantees.

Nearest-neighbor methods are *nonparametric*, in the sense that they rely on minimal assumptions on the data, namely, assuming only that "nearby points are likely to have similar labels."

The k-NN classifier and its variants are sufficiently flexible as to be able to learn *any* learnable dichotomy, in a well-defined sense. (The precise property is known as universal *Bayes consistency*, see below.) As the reader might well expect, learning the class of, say, all circles is much easier than learning arbitrary unstructured decision boundaries (see Fig. 1). An easy *no-free-lunch* theorem [58,

[1] Pelillo [49] traces the 1-NN method to Abū 'Alī al-Ḥasan ibn al-Ḥasan ibn al-Haytham (Alhazen)'s book *Kitāb al-Manāẓir* ("Book of Optics"), circa 1030.

A. Kontorovich (✉)
Department of Computer Science, Ben-Gurion University, Beer Sheva, Israel
e-mail: karyeh@cs.bgu.ac.il

S. Kpotufe
Department of Statistics, Columbia University, New York, NY, USA

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_5

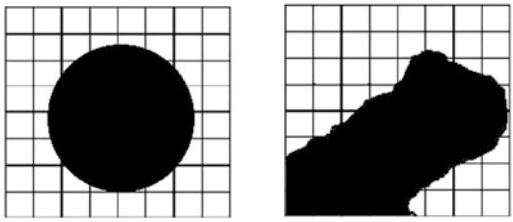


Fig. 1 Structured vs. unstructured decision boundaries (image on the right is based on [56])

Chapter 5] shows that this bias–complexity tradeoff is an inherent property of learnability: If one only needs to learn simple dichotomies, then a small number of examples will suffice, and conversely, arbitrarily complex dichotomies can require arbitrarily large sample sizes. Standard results based on VC theory (see, e.g., [58]) show that learning a sphere in d dimensions requires $O(d)$ examples. On the other hand, without additional smoothness assumptions, learning unstructured dichotomies such as in Fig. 1 requires roughly 2^d examples; this is the infamous *curse of dimensionality*.

This chapter aims at giving a high-level overview of modern metric-space techniques for classification and regression. Our goal is to provide an accessible, practically oriented survey, with pointers to the most recent literature on the subject.

1.1 Metric Spaces

NN methods fall under the category of *proximity*-based, and as such, they require a basic notion of distance. The latter can be formalized in several ways, perhaps the most natural being in terms of a *metric*.

Definition 1 A **metric space** $(\mathcal{X}, \rho)$ is a set $\mathcal{X}$ endowed with a *metric* $\rho : \mathcal{X}^2 \to [0, \infty)$, satisfying the three axioms: (i) positivity, $\rho(x, x') = 0 \iff x = x'$, (ii) symmetry, $\rho(x, x') = \rho(x', x)$, (iii) triangle inequality, $\rho(x, x') \leq \rho(x, x'') + \rho(x', x'')$ for all $x, x', x'' \in \mathcal{X}$.

Relaxing positivity to allow $\rho(x, x') = 0$ for $x \neq x'$ yields a **pseudo-metric**. Relaxing symmetry to allow $\rho(x, x') \neq \rho(x', x)$ yields a **quasi-metric** [65]. Allowing violations of the triangle inequality yields a **semi-metric** [66].

We warn the reader that these definitions are not entirely standard in the literature; our choice was based on the earliest sources we were able to locate.

Pseudo-metrics behave very similarly to ordinary metrics, and indeed, by collapsing all the $x' \in \mathcal{X}$ for which $\rho(x, x') = 0$ into an equivalence class, we recover the usual metric space over the equivalence classes (which may now be considered as points). Quasi- and semi-metrics, however, behave in ways starkly distinct from metrics, and very few learning-theoretic results are known for these. Two recent works exploring these venues include [29, 31]; these also contain other

references to the mathematical behavior of these metric spaces' more exotic cousins. This chapter will focus exclusively on metric spaces.

Definition 2 In any metric space $(\mathcal{X}, \rho)$, the (closed) **ball** center at $x \in \mathcal{X}$ with radius $r \geq 0$ is defined as $B(x, r) := \{x' \in \mathcal{X} : \rho(x, x') \leq r\}$. For $A \subset \mathcal{X}$, we write $B(A, r)$ to denote the r-**envelope** of A, namely $\cup_{x \in A} B(x, r)$. The **diameter** of $\mathcal{X}$ is the largest distance between any pair of points: $\operatorname{diam}(\mathcal{X}) = \sup_{x,x' \in \mathcal{X}} \rho(x, x')$.

Definition 3 (ℓ_p Norms) For $\mathcal{X} = \mathbb{R}^d$ and $p \in [1, \infty)$, the ℓ_p norm is defined by $\|x\|_p^p = \sum_{i \in [d]} |x_i|^p$. For $p = \infty$, we define $\|x\|_\infty = \max_{i \in [d]} |x_i|$. Any norm $\|\cdot\|$ induces a metric via $\rho(x, x') = \|x - x'\|$.

Assuming that our data reside in a metric space is less restrictive than requiring vectorial features. One can *always* impose a metric over a feature space (say, the Euclidean one), while some metrics are not induced by *any* norm.

Consider images. Although these can be naively represented as coordinate vectors in $\mathbb{R}^d$, the Euclidean (more generally, ℓ_p) distance between the representative vectors does not correspond well to the one perceived by human vision. Instead, the *earthmover* distance (EMD) is commonly used in vision applications [53]; in [27], a nearest-neighbor classifier was reported to achieve a 13% error on images using EMD, but a much worse 39% error using the Euclidean distance. Yet representing earthmover distances using any fixed ℓ_p norm unavoidably introduces very large inter-point distortion [47], potentially corrupting the data geometry before the learning process has even begun. Nor is this issue mitigated by kernelization, as kernels necessarily embed the data in a Hilbert space, again implying the above distortion—and so an approach of this type is ad hoc, precluding a principled treatment of non-Euclidean data. A similar issue arises for strings: these can be naively treated as vectors endowed with different ℓ_p metrics, but a much more natural metric over strings is the edit distance, which is similarly known to be strongly non-Euclidean [2].

Thus, in some sense, the NN methods we shall discuss are most interesting and relevant in the case of non-Euclidean—and even non-vectorial—metrics. The supremely important question of *how* to choose an appropriate metric for a given problem is beyond our scope. At some point, in any learning problem, one must appeal to domain knowledge, based on which features, generative models, and yes, metrics are suggested. Nor can there be any single "best" metric to use in every case; this is entailed by the classic no-free-lunch theorem, which states that no learning algorithm can uniformly outperform every other one [58, Theorem 5.1].

Since we are going to be discussing metric spaces more general than the familiar Euclidean space $(\mathbb{R}^d, \|\cdot\|_2)$, we will generalize the notion of *dimension* appropriately. While numerous definitions of dimension have been proposed for general metric spaces, for algorithmic purposes, the *doubling dimension* is by far the most common one (starting, apparently, with [14]).

Definition 4 (Doubling Dimension) Suppose that $(\mathcal{X}, \rho)$ is a metric space such that every ball of radius b can be covered by N balls of radius $b/2$; moreover, let $N = N(\mathcal{X}, \rho)$ be the smallest such number. The **doubling dimension** of $(\mathcal{X}, \rho)$ is defined as $\operatorname{ddim}(\mathcal{X}, \rho) = \log_2 N$.

It is straightforward to show that $\operatorname{ddim}(\mathbb{R}^d, \ell_p) = d + O(1)$ for all d and p. When the earthmover distance (EMD, mentioned above) is computed over multisets of size k, its doubling dimension is $O(k \log k)$ [27, Lemma 5]. A metric commonly used in the context of time series is edit distance with real penalty (ERP) [13]. When restricted to k-sparse time-series vectors, ERP has doubling dimension $O(k \log k)$ [27, Lemma 7].

Definition 5 For $\gamma > 0$, a γ-**cover** of a metric space $(\mathcal{X}, \rho)$ is a set $A \subset \mathcal{X}$ such that $B(A, \gamma) \supseteq \mathcal{X}$ (in words: the γ-envelope of A contains all of $\mathcal{X}$). A γ-**packing** of $\mathcal{X}$ is an $A \subset \mathcal{X}$ that is γ-separated: for distinct $x, x' \in A$, we have $\rho(x, x') > \gamma$. Finally, a γ-**net** of $\mathcal{X}$ is a set A that is simultaneously a covering *and* a packing; as such, it is readily verified to be a *minimal* cover and a *maximal* packing.[2]

In a doubling space, the size of any γ-net is bounded by $\left(\frac{2\operatorname{diam}(\mathcal{X})}{\gamma}\right)^{\operatorname{ddim}(\mathcal{X})}$ [27].

1.2 Learning Framework

In learning-theoretic terminology [46], our *instance space* will always be some metric space $(\mathcal{X}, \rho)$, often a Euclidean metric for illustration. Additionally, we have a *label space* $\mathcal{Y}$, which will be either a discrete set of categories (for classification) or a subset of $\mathbb{R}$, for regression. We assume a joint distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, unknown to the learner; this is the standard and natural *agnostic learning* setting. The learner receives a sequence of n instance–label pairs, (X_i, Y_i), drawn i.i.d. from $\mathcal{D}$, based on which he constructs a predictor (i.e., classifier or regressor) $\hat{f} : \mathcal{X} \to \mathcal{Y}$. A *loss function* $\ell : \mathcal{Y}^2 \to [0, \infty)$ quantifies the quality of a prediction. For classification, the zero-one loss $\ell(y, y') = \mathbb{1}\left[y \neq y'\right]$ is typically chosen; for regression, the square loss $\ell(y, y') = (y - y')^2$ is common. The *risk* of any $g : \mathcal{X} \to \mathcal{Y}$ is defined as

[2] Minimal/maximal in the sense that no points can be added to (resp., removed from) A without violating the covering (resp., packing) property.

$$R(g) = \mathop{\mathbb{E}}_{(X,Y)\sim\mathcal{D}}[\ell(g(X), Y)].$$

A minimizer f^* of $R(\cdot)$ over all g is called a Bayes-optimal predictor, and its risk $R^* := R(f^*)$ is called the Bayes-optimal risk. The learning algorithm mapping samples to predictors is said to be Bayes-consistent if $R(\hat{f}) \to R^*$ as $n \to \infty$.[3] We also define the *empirical risk* of a predictor, $R_n(g) = \frac{1}{n}\sum_{i=1}^{n} \ell(g(X_i), Y_i)$. Finally, the *excess risk* is $R(g) - R^*$.

1.2.1 Consistency and Typical Rates

Although the 1-NN classifier is not in general Bayes-consistent [15], taking a majority vote among the k-nearest neighbors does guarantee Bayes consistency, provided that k increases appropriately in sample size [59, 20, 69].

Given training data $(X_i, Y_i)_{i\in[n]}$, one evaluates the k-NN classifier $g_{k-\mathrm{NN}}$ at a test point x by sorting[4] the data in increasing order of distance from x, say as $(X_{(1)}, Y_{(1)}), (X_{(2)}, Y_{(2)}), \ldots, (X_{(n)}, Y_{(n)})$. Only the first k pairs are considered, and among these, the majority label[5] is predicted. See Eqs. 1 and 2 for more formal definitions.

Let k be fixed for now. For metric spaces with finite diameter and doubling dimension d, under some "niceness" assumptions on the distribution, the expected excess risk of the k-NN classifier decays as $O(R^*/\sqrt{k} + (k/n)^{1/d})$ ([8, 22] for Euclidean spaces and [11] for doubling metrics). Analogous rates hold for regression under the ℓ_2 metric. See Sect. 2 for a more in-depth discussion of the various aspects of a data distribution, metric choice, and the number of neighbors that affect performance in regression and classification.

Since R^* and other distribution-dependent parameters are unknown to the learner, these bounds provide the practitioner neither with estimates of the empirical risk R_n nor a method for choosing k; the latter is achieved in practice via cross-validation. Any of the approximate-NN techniques mentioned in Sect. 1.3 can be used to speed up k-NN evaluation (with some pre-processing cost). Some of these approximate-NN methods maintain Bayes consistency and even the risk convergence rate (or nearly so), see, e.g., [23, 68].

[3] Our definition was not rigorous since the mode of convergence was not specified; see [38] for a rigorous definition as well as a complete characterization of the metric spaces that admit such a learner.

[4] The question of how to break ties is handled by various theoretical analyses but almost never comes up when processing real-world data, so we shall ignore it here.

[5] or *plurality*, in the multiclass case.

1.3 Computational Efficiency

Until recently, research on NN-based methods tended to focus somewhat dichotomously on either the statistical or the computational aspects. On the statistical front, the most commonly investigated questions involve Bayes consistency and rates of convergence under various distributional assumptions [11, 19, 21, 24, 33, 37, 44]. In classical k-NN-based learning, the training phase involves choosing the k—the number of neighbors—via cross-validation, or might be trivial if we view k as fixed (just store the n labeled examples), and *brute-force* evaluation (searching for neighbors among n datapoints) takes time proportional to $n\tau$, where τ is the time it takes to evaluate $\rho(x, x')$. For the ℓ_p metrics, τ is $O(d)$, which for large n and d results in an expensive evaluation time of $O(nd)$.

An orthogonal body of literature developed a host of techniques for fast prediction in time considerably better than linear in sample size n. These are the so-called *fast proximity search* procedures whose main aim is to quickly return any number of neighbors of a query x out of a database of n training points [1, 6, 28, 16, 40, 10, 54]. We especially recommend the comprehensive survey [57].

A major bottleneck remains, as the number k of desired neighbors can itself be large—e.g., as a function of desired accuracy, unless various data quantization and subsampling techniques are employed on top of fast search: here, the idea is to return neighbors out of a *compressed* version of the original dataset.

These considerations are discussed further in Sects. 3 and 4, including recent insights on the types of tradeoffs on accuracy that might be induced by fast proximity search and data compression techniques.

2 Vanilla Nearest-Neighbor Prediction

Here we consider nearest-neighbor methods for classification and regression. As it turns out, classification is quite related to regression (see Sect. 2.1), and we can therefore give a combined treatment that yields insights into desirable properties of the distance function $\rho(x, x')$ and proper choices of k (the number of neighbors). These insights then lead to more modern implementations that aim at: (1) improving or maintaining prediction accuracy, while (2) decreasing computation time.

We need a bit of formalism for the discussion that follows. We have assumed so far that the input variable X belongs to a space $\mathcal{X}$ endowed with a distance function $\rho(x, x')$, perhaps a metric (see Sect. 1.1). The problem in either regression or classification is to predict the label Y of X, where in regression, Y is assumed to be a vector in $\mathcal{Y} \subset \mathbb{R}^L$ (most often Y is a scalar, i.e., $L = 1$), while in classification, Y is assumed to take on one of the L possible classes, say $\mathcal{Y} \doteq \{1, \ldots, L\}$.

Given a sample $\{(X_i, Y_i)\}_{i=1}^n$, the aim is to construct (i.e., learn) a predictor that maps any value $x \in \mathcal{X}$ to some label $Y \in \mathcal{Y}$, based on (the labels of k) nearest neighbors of x in $\{X_i\}_1^n$. In all that follows, we let $k\mathrm{NN}(x)$ denote the k closest

datapoints in $\{X_i\}_1^n$ to x under the distance ρ. We assume for simplicity that ties are broken arbitrarily or that more than k datapoints are returned in $k\text{NN}(x)$ in case of ties. In practice, ties usually do not make a big difference as the most important choices will be that of ρ and k. We will henceforth assume that exactly k points are returned for any query $x \in \mathcal{X}$.

Definition 6 (k-NN Regression) Assume $\mathcal{Y} \subset \mathbb{R}^L$ and that $Y = f(X) + \text{noise}$ for some unknown *regression function* f and unknown noise variable (usually assumed independent of X and of 0 mean). A k-NN regressor $\hat{f}_k$ based on n samples is then defined as the average Y values of $k\text{NN}(x)$

$$\hat{f}_k(x) \doteq \frac{1}{k} \sum_{X_i \in k\text{NN}(x)} Y_i. \tag{1}$$

The hope is that $\hat{f} \approx f$ for n sufficiently large; in other words, the regressor performs a *denoising* function. We will later discuss how choices of ρ and k influence the quality of estimation.

Classification is similar, where we replace the above average by a majority label.

Definition 7 (k-NN Classification) Let $\mathcal{Y} \doteq \{1, \ldots, L\}$. The k-NN classifier $\hat{h}_k$ is then defined as

$$\hat{h}_k(x) = \text{Majority } Y \text{ value over } k\text{NN}(x). \tag{2}$$

As majority are related to averages, we relate $\hat{h}_k$ to $\hat{f}_k$ in the next section, the importance being that regression error influences classification performance.

2.1 A Link Between Regression and Classification

The classifier $\hat{h}_k$ can be related to its regression counterpart as follows. Again, let the labels $Y \in \mathcal{Y} \doteq \{1, \ldots, L\}$, and now for any label Y corresponding to X, define the encoding vector $\tilde{Y}$ where $\tilde{Y}^l = \mathbb{1}\{Y = l\}$. Define the regressor $\hat{f}_k(x)$ as averaging $\tilde{Y}$ labels of neighbors of x; in other words, $\hat{f}_k(x)$ is a vector in $\mathbb{R}^L$ whose coordinate $\hat{f}_k^l(x)$, $l \in [L]$ is exactly the proportion of label l among the $k\text{NN}(x)$. Thus $\hat{h}_k$ is just identifying the maximum coordinate of $\hat{f}_k$, i.e.,

$$\hat{h}_k(x) = \arg\max_{l \in [L]} \hat{f}_k^l(x).$$

Note that the regressor $\hat{f}_k(x)$ is estimating the *regression function* $f(x) \in \mathbb{R}^L$ whose coordinate $f^l(x) \doteq \mathbb{P}\,|\,(Y = l \mid X = x)$, where we assume that data are drawn from some unknown joint distribution on X, Y, allowing nondeterministic label values Y at any query x. Nondeterminism accounts for mistakes or uncertainty in

labels, e.g., mistakes or disagreements in labeling objects in images, or uncertain treatment effects in medical or financial applications. In the cases where Y is deterministic (or nearly so) at any query x, $f^l(x) = 1$ for the true label l of x and otherwise equals 0 for all other coordinates $l \neq l$.

Note that the best possible classification at x, the so-called Bayes classification, is given by $h^*(x) = \arg\max_{l \in [L]} f^l(x)$, since this choice (the most expected label at x) minimizes the classification error (probability of misclassification at x).

It follows that regression performance influences classification performance. However, classification can be much easier than regression in terms of sample size required for good performance.

If $\hat{f}_k(x)$ is close to $f(x)$, i.e., sample proportions of labels are close to true population proportions in the vicinity of a query x, it is then likely that the majority label $\hat{h}_k$ generalizes to the best population choice $h^*(x) = \arg\max_{l \in [L]} f^l(x)$. However, if there is *a large margin* between population proportions of labels, e.g., $f^l(x) = \mathbb{P}(Y = 1|x)$ is large for some l, then this can be quickly discovered with few samples without estimating f that well.

Such fact is used in early analysis and insights on k-NN classification, arguably the most common form of prediction task in machine learning applications. In fact, one can show that classification error is upper-bounded by regression error, suggesting that classification can be at times as hard as regression, but never harder (see, e.g., Theorem 6.5 in [18]). In particular, such relation largely drives early insights on choices of ρ and k that lead to best prediction performance for either classification or regression.

2.2 Choice of Distance $\rho(x, x')$

2.2.1 Choices of ρ in Practice

First, we note that choice of a distance measure ρ is often interchangeable with choices of data representation, e.g., dimension reduction, feature selection, embeddings, or more modern representations learned from neural nets. In other words, using Euclidean distance after a choice of data representation $x \mapsto \Phi(x)$ boils down to a choice of distance $\rho(x, x') = \|\Phi(x) - \Phi(x')\|$. On the other hand, most common distances ρ, e.g., weighted Euclidean distances $\rho^2(x, x') = (x - x')^\top W(x - x')$, can be reduced to a choice of data representation $\Phi(x) = W^{1/2}(x - x')$.

Despite this connection between choices of data representation and choices of distances ρ, the two problems tend to be studied separately. In the case of data representation, the field of *representation learning* is concerned with estimating

data representations $\Phi(x)$ under which simple classifiers (e.g., linear classifiers, nearest neighbors) perform well, ideally across multiple related tasks [5, 51, 60]. Quite remarkable in many applications (e.g., computer vision, speech) is the fact that generic representations given by layers of neural networks result in the best performance achievable by k-NN methods—among other simple classifiers—over most other reasonable choices of metrics. There is at the moment no definitive explanation for such successful representation, beyond general insights about favorable metrics (see rest of this section).

The area of the so-called *metric learning* is most directly concerned with estimating distances ρ under which vanilla nearest-neighbor methods perform well. For $x \in \mathbb{R}^d$, metric learning generally boils down to optimizing some objective $O(W)$ over all weighted-Euclidean metrics of the form $\rho^2(x, x') = (x-x')^\top W(x-x')$, for a PSD matrix W. Generally, the objective $O(W)$ is a convex surrogate for the classification loss of nearest-neighbor methods $\hat{f}_k$ based on ρ [67, 26, 17, 61]. These approaches are somewhat limited by the choice of *optimizable* metrics such as above, at least relative to general representation learning, although there are clear situations where weighted-Euclidean metrics are ideal (see below, including discussions of Fig. 2).

2.2.2 Theoretical Insights on Choices of ρ

To understand desirable properties of the distance $\rho(x, x')$, we mainly draw insight from regression, since as argued above, good regression performance implies good classification performance. As such, we let $\hat{f}_k(x)$ denote the regressor of (1) defined over labels $Y \in \mathbb{R}^L$ in case of regression, or defined over $\tilde{Y} \in \mathbb{R}^L$ in case of

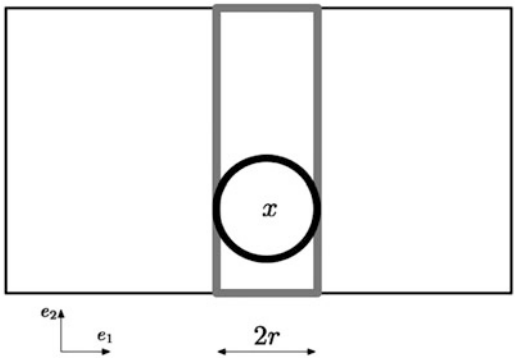


Fig. 2 Two different distances ρ on $\mathcal{X} \in \mathbb{R}^2$ (shown are corresponding *balls* $B(x, r) \doteq \{x' : \rho(x, x') \leq r\}$). The first is just Euclidean distance $\rho(x, x') \doteq \|x - x'\|$, with $B(x, r)$ given as the black circle. The second is the distance along coordinate direction e_1, i.e., $\rho(x, x') \doteq |x^{(1)} - x'^{(1)}|$. The corresponding ball $B(x, r)$ is given by the more massive gray rectangle. Intuitively, the first induces a space of dimension 2, while the second induces a space of lower dimension 1

classification (see previous Sect. 2.1). Similarly, we let the unknown regression function $f(x)$ correspond to $\mathbb{E}\,|\,[Y \mid x]$ or $\mathbb{E}\,[\tilde{Y} \mid x]$, respectively, for regression or classification problems.

Desirable properties of ρ are two-fold, corresponding to mitigating the sources of error in estimating the regression function $f(x)$ by $\hat{f}_k(x)$, namely errors contributed by the *variance* and *bias* of the estimate $\hat{f}_k(x)$.

Bias and Variance of $\hat{f}_k$ For simplicity, let us view the input variables $\{X_i\}$ as fixed and only consider the randomness in the output $\{Y_i\}$. Then, since $\mathbb{E}\,|\,[Y_i \mid X_i]$ (or resp., $\mathbb{E}\,[\tilde{Y}_i \mid X_i]$ in classification) equals $f(X_i)$, we might view $\hat{f}_k(x)$ as estimating $\mathbb{E}\,\hat{f}_K(x) = \frac{1}{k}\sum_{X_i \in k\mathrm{NN}(x)} f(X_i)$, i.e., the average f value among $k\mathrm{NN}(x)$. Let us denote this $\tilde{f}(x; k\mathrm{NN}(x))$, using the notation[6] $\tilde{f}(x, B) \doteq \mathbb{E}\,|\,[f(X) \mid X \in B]$. Since we expect $\hat{f}_k(x)$ to be close to $\tilde{f}(x; k\mathrm{NN}(x))$, we can view the error $|\hat{f}_k(x) - f(x)|$ as the sum of two terms: a so-called *variance* term $|\hat{f}_k(x) - \tilde{f}(x; k\mathrm{NN}(x))|$ (how well we estimate the average value of f in a neighborhood of x) and a *bias* term $|\tilde{f}(x; k\mathrm{NN}(x)) - f(x)|$ (how close the average value of f in a neighborhood of x is to $f(x)$).

This leads to the following desirable properties of ρ, w.r.t. to the joint distribution $\mathcal{D}_{X,Y}$ of the data:

Under the ideal choice of ρ, (i) values of f (i.e., expected Y labels) should not change much over points X_i's that are close to a query x and (ii) many datapoints X_i's should fall close to typical queries x.

(i) **Closeby points are expected to have similar Y values** (low bias). Intuitively, nearest-neighbor methods operate under the assumption that expected labels—determined by f—do not change much in *small* neighborhoods of a query x, where neighborhoods are determined by the choice of distance ρ. This can be encoded in a variety of ways.
One way is to require that $f(x) \approx f(x')$ whenever $\rho(x, x') \approx 0$: these are various smoothness assumptions (e.g., Lipschitz or Hölder f) present in traditional analyses of k-NN [34, 58].
A second way most directly integrates with nearest-neighbor estimates $\hat{f}_k(x)$ (which average Y values over neighbors of x). Namely, one requires that the

[6] This is an abuse of notation, since the expectation in the case of $\tilde{f}(x; k\mathrm{NN}(x))$ is over the empirical distribution on $\{X_i\}$, but for general $B \subset \mathcal{X}$ is over the data distribution P_X.

average of f over any small neighborhood of x stays close to $f(x)$; that is, $f(x)$ is close to $\tilde{f}(x; B_r(x)) \doteq \mathbb{E}\,|\,[f(X) \mid X \in B_r(x)]$ for neighborhoods $B_r(x) \doteq \{x' : \rho(x, x') \leq r\}$, when r is small [33].

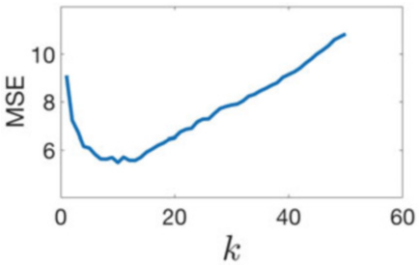
(ii) **Typical queries x have many close neighbors** (low variance + low bias). The estimate $\hat{f}_k(x)$ is being an average over k values, and it is desirable that we can choose large k to decrease its variance. However, the larger k, the farther the neighbors in $k\text{NN}(x)$ might be from x in ρ-distance; we therefore want it to be that ρ and the data distribution $\mathcal{D}_X$ allow for many neighbors of x that are close to x under ρ. In other words, $\mathcal{D}_X(B(x, r))$ is relatively large for small r. Again there are many ways to encode such desired property of $(\mathcal{D}_X, \rho)$, all of which can be viewed as notions of *intrinsic dimension* of the pair $(\mathcal{D}_X, \rho)$.

For intuition, consider the fact that the volume of a Euclidean ball $B(x, r)$ behaves like $O(r^d)$, where we consider small $r \approx 0$; such volume is largest for small values of d. We can hence view the desirables above as requiring that the pair $(\mathcal{D}_X, \rho)$ induces a low-dimensional data space. This is illustrated in Fig. 2.

In the example of Fig. 2, the metric $\rho(x, x') \doteq |x^{(1)} - x'^{(1)}|$ is the limit of weighted Euclidean distances giving small to zero weight to coordinate e_2; such distances would typically improve variance over a Euclidean choice (desirable (ii)) and would be preferred for instance if the unknown f has little to no dependence on e_2 since then ρ also satisfies (i) whenever the Euclidean distance does.

Finally, we note that recently [11] showed that the two desired properties above, (i) and (ii), can be captured at once by relating local changes $|f(x) - \tilde{f}(x; B(x, r))|$ to the mass $\mathcal{D}_X(B(x, r))$ of small neighborhoods $B(x, r)$.

2.3 *Choice of $k = k(n)$*



Assuming a good distance choice ρ, the choice of number of neighbors k is crucial to performance. This is easy to understand by building on the above discussion of bias and variance of $\hat{f}_k$: smaller values of k (down to $k = 1$) ensure that nearest neighbors of a query x are as close as possible to x (as permitted by ρ) and hence induce small bias but large variance, while large values of k induce

low variance but potentially high bias. This results in a situation where, while k increases, the error $|\hat{f}_k(x) - f(x)|$ decreases at first and then increases again when bias dominates variance (such behavior is illustrated in the wrapped figure).

A good choice of k should be made by cross-validation on $k \in [n]$. The best possible choice is generally an increasing function of sample size n.

The right choice of k is affected by the interaction between data distribution and choice of ρ. In particular, the following two aspects of a $\mathcal{D}, \rho$ have a strong influence.

Effects of Intrinsic Dimension In higher-dimensional spaces, datapoints tend to be disperse, and we therefore expect smaller choices of k to perform better by mitigating bias. For instance, suppose $\mathcal{D}$ is uniform on a D-dimensional hypercube, and then the k-nearest neighbor of a query x is expected to be at distance $O(k/n)^{1/D}$, which grows exponentially with dimension D (a *curse of dimension*).

Fortunately, it is often the case that high-dimensional data in $\mathbb{R}^D$ lie close to a smaller d-dimensional space, or could be highly sparse (i.e., most coordinates are 0), or as discussed above, the distance ρ might effectively induce a lower-dimensional space (as in the example of Fig. 2). In these cases, while the data appear high-dimensional, nearest-neighbor distances are small, conforming to the *intrinsic dimension of the problem*, and resulting in choices of k with good performance tradeoffs.

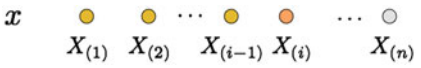
Nearest-neighbor methods are adaptive to unknown intrinsic dimension.

Local Choices of $k = k(x, n)$ It is understood that performance can be further improved by choosing k as a function of the query point x; here we cannot pick $k(x)$ by cross-validation since we do not observe the error at x. However, many approaches have been proposed [4, 24, 37, 45], which aim to balance the variance of $\hat{f}_k(x)$ using sample-based indicators such as distances to neighbors, the local mass of points, and the margin between estimated proportions of classes. Such approaches can, however, be relatively more expensive at prediction time (already affected by the search for neighbors), since they involve more decisions per query x.

3 Distributed 1-NN vs. Weighted k-NN

Prediction time can be expensive with k-NN methods, especially when large k is chosen (to improve prediction performance as discussed above). On the other hand, 1-NN, i.e., where $k = 1$, yields much faster prediction, albeit at the cost of decreased accuracy. A simple way to obtain a good tradeoff is to aggregate multiple 1-NN predictors, each defined on a random subsample S_i of the original data $\{(X_i, Y_i)\}_{i=1}^n$. For classification, aggregation consists of taking the majority of the predicted classes, while for regression we simply take the average of the predicted values. The sub-sampled 1-NN predictors can be estimated in parallel resulting in fast prediction time with respect to k-NN. The approach, often called *bagging* of 1-NNs, achieves better prediction accuracy than 1-NN, almost as good of that of k-NN, as the number of subsamples increases (see Fig. 3).

The improved accuracy over vanilla 1-NN, almost on par with k-NN, can be explained as follows: bagging of 1-NNs can be viewed as a form of *weighted* k-NN, i.e., as the number of subsamples increases, it accounts for all k neighbors of a query x but assigns decreasing weights to the farther neighbors. Consider the figure below, where a query x is shown along with datapoints in order of distance from x.



Then, the main insight is that, if N subsamples are used (each of size say $m \ll n$), the ith nearest neighbor $X_{(i)}$ appears as the closest neighbor of x in a proportion p_i of subsamples, p_i decreasing with i. This is because p_i is essentially the likelihood that none of the closer neighbors of x in the initial sample $(X_{(1)}, \ldots, X_{(i-1)})$ appears in a random subsample; that is, $p_i \approx (1 - \frac{m}{n})^{i-1}$, where $\frac{m}{n}$ is approximately the

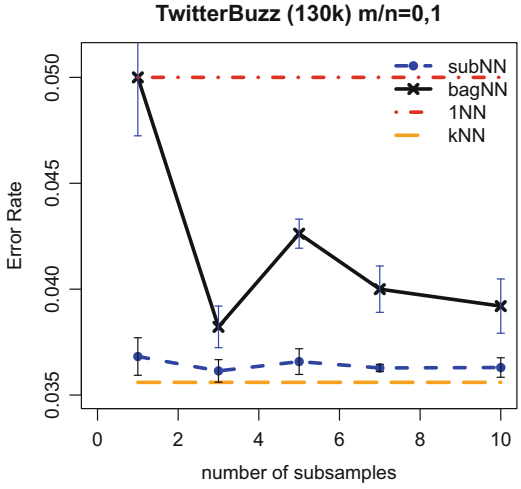


Fig. 3 Predicting viral tweets using aggregate 1-NNs over distributed subsamples. The vanilla approach is labeled above as bagNN, while subNN is a variant of [68] that denoises the data first

likelihood of picking any given datapoint to form a subsample of size m. This intuition holds as $N \to \infty$ as formalized, e.g., in [7, 55]. Further denoising of the original data, i.e., replacing labels Y_i by more stable labels Y_i' learned in a first prediction pass, further improves accuracy over future queries x [68, 50].

Rather than using *random* subsampling, the compression-based approach via γ-nets discussed above might yield improved performance.

4 Compression-Based 1-NN

An alternative to choosing the subsample randomly, as discussed in Sect. 3, is to select it with some explicit "coverage" criterion in mind. In benign cases, it will be possible to retain a significantly reduced fraction of the full training sample while preserving much of the relevant information in the original sample. A simple, intuitive (and, as it turns out, mathematically well-motivated!) way of formalizing the quality of a subsample is to examine the accuracy of the 1-NN classifier it induces. Indeed, any labeled set of points in a metric space induces a 1-NN classifier in the same manner as the full sample would. If this "compressed" classifier achieves a high agreement on the full sample (i.e., a low training error), then standard bounds in the spirit of *Occam's razor* (discussed below) guarantee a good generalization performance.

The problem of computing a minimal subsample of a larger labeled sample whose induced 1-NN classifier achieves zero training error was historically known as *sample condensing*. Hart [39] proposed a greedy sample condensing heuristic with runtime $O(n^3)$ and no guarantees on optimality (i.e., the relation between the obtained compression set size to the optimal one). The runtime was later improved by [3] to $O(n^2)$—and in fact, an extensive research was devoted to studying the sample condensing problem [25, 52, 64]—again with no optimality guarantees. This problem was shown to be NP-hard [63, 70], and a hardness-of-approximation result was given by [30]. The latter paper also gave an algorithm with runtime achieving nearly the best possible (assuming P $\neq$ NP), more on which below.[7]

Leveraging the results of [30], a compression-based alternative to k-NN has been developed in the series of works [27, 43, 41]. This methodology enjoys a number of statistical and computational advantages over the traditional k-NN classifier. Salient among these are explicit data-dependent generalization bounds, and considerable runtime and memory savings. Informally, a learning algorithm is a *compression scheme* of size κ if the predictor it constructs is fully determined by some subset of κ labeled examples from the labeled sample; the remaining examples may be discarded.[8] For example, if running hard SVM on a dataset results in κ support

[7] In a follow-up work, [29] showed that the relaxed problem of finding a small *nearly* consistent compression set is still hard to approximate, under standard complexity assumptions.

[8] Some additional bits of *side information* are also allowed, see [41, Section 6.1].

vectors, these uniquely define the separating hyperplane. Very roughly speaking, any classifier g induced by a compression scheme of fixed size κ satisfies $R(f) \leq R_n(f) + \sqrt{\frac{\kappa \log n}{n}}$ with high probability. An exact statement may be found in [32, Theorem 2] and a refined "fast rate" version in [29, Theorem 8].

Since *any* choice of κ labeled examples uniquely determines a 1-NN classifier, the challenge is to choose this subset wisely. A fruitful idea for the choice of the compression set, pursued in [27, 43, 30, 41] and related works, is that of a γ-net. The latter can always be constructed in time $O(n^2)$ on an n-point sample, but in doubling spaces, a runtime of $2^{O(\mathrm{ddim}\mathcal{X})} n \log \frac{1}{\gamma}$ can be achieved. The resulting compression set, a γ-net, has size $\left(\frac{\mathrm{diam}(\mathcal{X})}{\gamma}\right)^{O(\mathrm{ddim}(\mathcal{X}))}$. Perhaps surprisingly, it turns out that achieving significantly smaller compression sets is NP-hard—so γ-net-based compression is, in some sense, nearly optimal. See [30] for details, including algorithms and proofs of the aforementioned claims. One can tune γ by cross-validation or by performing a structural risk minimization on a bound consisting of a sample error and a complexity term, as in [29]. The compression-based approach turns out to be more robust than k-NN-based classification, in the sense that it is Bayes-consistent for every metric space and every distribution that admit *any* such learner [36, 38, 42].

References

1. Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.
2. Alexandr Andoni and Robert Krauthgamer. The computational hardness of estimating edit distance. *SIAM J. Comput.*, 39(6):2398–2429, April 2010.
3. Fabrizio Angiulli. Fast condensed nearest neighbor rule. In *ICML*, pages 25–32, 2005.
4. Akshay Balsubramani, Sanjoy Dasgupta, Yoav Freund, and Shay Moran. An adaptive nearest neighbor rule for classification. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada*, pages 7577–7586, 2019.
5. Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.
6. Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *ICML*, pages 97–104, New York, NY, USA, 2006. ACM.
7. Gérard Biau, Frédéric Cérou, and Arnaud Guyader. On the rate of convergence of the bagged nearest neighbor estimate. *Journal of Machine Learning Research*, 11(Feb):687–712, 2010.
8. Gérard Biau and Luc Devroye. *Lectures on the nearest neighbor method*. Springer Series in the Data Sciences. Springer, Cham, 2015.
9. Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

10. Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19–21, 2002, Montréal, Québec, Canada*, pages 380–388, 2002.
11. K. Chaudhuri and S. Dasgupta. Rates of convergence for nearest neighbor classification. In *Advances in Neural Information Processing Systems*, 2014.
12. George H. Chen and Devavrat Shah. Explaining the success of nearest neighbor methods in prediction. *Foundations and Trends®in Machine Learning*, 10(5–6):337–588, 2018.
13. Lei Chen and Raymond Ng. On the marriage of Lp-norms and edit distance. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 792–803. VLDB Endowment, 2004.
14. Kenneth L. Clarkson. Nearest neighbor queries in metric spaces. *Discrete Comput. Geom.*, 22(1):63–93, 1999.
15. Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.
16. Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 253–262. ACM, 2004.
17. Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 209–216. ACM, 2007.
18. L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1997.
19. Luc Devroye. On the inequality of Cover and Hart in nearest neighbor discrimination. *IEEE Trans. Pattern Anal. Mach. Intell.*, 3(1):75–78, 1981.
20. Luc Devroye and László Györfi. *Nonparametric density estimation: the L_1 view*. Wiley Series in Probability and Mathematical Statistics: Tracts on Probability and Statistics. John Wiley & Sons, Inc., New York, 1985.
21. Luc Devroye, László Györfi, Adam Krzyżak, and Gábor Lugosi. On the strong universal consistency of nearest neighbor regression function estimates. *Ann. Statist.*, 22(3):1371–1385, 1994.
22. Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31 of *Applications of Mathematics (New York)*. Springer-Verlag, New York, 1996.
23. Klim Efremenko, Aryeh Kontorovich, and Moshe Noivirt. Fast and Bayes-consistent nearest neighbors. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2020.
24. Sébastien Gadat, Thierry Klein, and Clément Marteau. Classification in general finite dimensional spaces with the k-nearest neighbor rule. *Ann. Statist.*, 44(3):982–1009, 06 2016.
25. W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18:431–433, 1972.
26. Amir Globerson and Sam T Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems*, pages 451–458, 2006.
27. Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. Efficient classification for metric data. *IEEE Transactions on Information Theory*, 60(9):5750–5759, 2014.
28. Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. Adaptive metric dimensionality reduction. *Theoretical Computer Science*, pages 105–118, 2016.
29. Lee-Ad Gottlieb, Aryeh Kontorovich, and Pinhas Nisnevitch. Nearly optimal classification for semimetrics. *Journal of Machine Learning Research*, 2017.
30. Lee-Ad Gottlieb, Aryeh Kontorovich, and Pinhas Nisnevitch. Near-optimal sample compression for nearest neighbors. *IEEE Trans. Information Theory*, 64(6):4120–4128, 2018.
31. Lee-Ad Gottlieb and Shira Ozeri. Classification in asymmetric spaces via sample compression. *CoRR*, abs/1909.09969, 2019.
32. Thore Graepel, Ralf Herbrich, and John Shawe-Taylor. PAC-Bayesian compression bounds on the prediction error of learning algorithms for classification. *Machine Learning*, 59(1–2):55–76, 2005.

33. L. Gyorfi. The rate of convergence of k_n-nn regression estimates and classification rules. *IEEE Transactions on Information Theory*, 27(3):362–364, 1981.
34. L. Györfi, M. Kohler, A. Krzyzak, and H. Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.
35. László Györfi, Michael Kohler, Adam Krzyżak, and Harro Walk. *A distribution-free theory of nonparametric regression*. Springer Series in Statistics. Springer-Verlag, New York, 2002.
36. László Györfi and Roi Weiss. Universal consistency and rates of convergence of multiclass prototype algorithms in metric spaces. *CoRR*, abs/2010.00636, 2020.
37. Peter Hall, Byeong U. Park, and Richard J. Samworth. Choice of neighbor order in nearest-neighbor classification. *The Annals of Statistics*, 36(5):2135–2152, 2008.
38. S. Hanneke, A. Kontorovich, S. Sabato, and R. Weiss. Universal Bayes consistency in metric spaces. *to appear in Ann. Stat.*, 2021+.
39. Peter E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.
40. Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 604–613. ACM, 1998.
41. Aryeh Kontorovich, Sivan Sabato, and Ruth Urner. Active nearest-neighbor learning in metric spaces. *Journal of Machine Learning Research*, 18:195:1–195:38, 2017.
42. Aryeh Kontorovich, Sivan Sabato, and Roi Weiss. Nearest-neighbor sample compression: Efficiency, consistency, infinite dimensions. In *Advances in Neural Information Processing Systems 30*, pages 1572–1582, 2017.
43. Aryeh Kontorovich and Roi Weiss. Maximum margin multiclass nearest neighbors. In *ICML*, 2014.
44. Samory Kpotufe. Fast, smooth and adaptive regression in metric spaces. In *Advances in Neural Information Processing Systems 22*. 2009.
45. Samory Kpotufe. k-NN regression adapts to local intrinsic dimension. In *Advances in Neural Information Processing Systems 24*, pages 729–737, 2011.
46. Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.
47. Assaf Naor and Gideon Schechtman. Planar earthmover is not in l_1. *SIAM J. Comput.*, 37:804–826, June 2007.
48. Haukur Pálmason, Björn Þór Jónsson, Laurent Amsaleg, Markus Schedl, and Peter Knees. On competitiveness of nearest-neighbor-based music classification: A methodological critique. In Christian Beecks, Felix Borutta, Peer Kröger, and Thomas Seidl, editors, *Similarity Search and Applications*, pages 275–283, Cham, 2017. Springer International Publishing.
49. Marcello Pelillo. Alhazen and the nearest neighbor rule. *Pattern Recognition Letters*, 38:34–37, 2014.
50. Xingye Qiao, Jiexin Duan, and Guang Cheng. Rates of convergence for large-scale nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 10768–10779, 2019.
51. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings*, 2016.
52. G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21:665–669, 1975.
53. Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.
54. Hanan Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.
55. Richard J Samworth et al. Optimal weighted nearest neighbour classifiers. *The Annals of Statistics*, 40(5):2733–2763, 2012.

56. C. Scott and R.D. Nowak. Minimax-optimal classification with dyadic decision trees. *IEEE Transactions on Information Theory*, 52(4):1335–1353, 2006.
57. G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing series)*. The MIT Press.
58. Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.
59. Charles J. Stone. Consistent nonparametric regression. *The Annals of Statistics*, 5(4):595–620, 1977.
60. Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International Conference on Machine Learning*, pages 1083–1092, 2015.
61. Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1473–1480, 2006.
62. Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.
63. Gordon Wilfong. Nearest neighbor problems. In *Proceedings of the Seventh Annual Symposium on Computational Geometry*, SCG '91, pages 224–233, 1991.
64. D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38:257–286, 2000.
65. Wallace Alvin Wilson. On Quasi-Metric Spaces. *Amer. J. Math.*, 53(3):675–684, 1931.
66. Wallace Alvin Wilson. On Semi-Metric Spaces. *Amer. J. Math.*, 53(2):361–373, 1931.
67. Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, pages 521–528, 2003.
68. Lirong Xue and Samory Kpotufe. Achieving the time of 1-nn, but the accuracy of k-nn. pages 1628–1636, 2018.
69. Lin Cheng Zhao. Exponential bounds of mean error for the nearest neighbor estimates of regression functions. *J. Multivariate Anal.*, 21(1):168–178, 1987.
70. A. V. Zukhba. Np-completeness of the problem of prototype selection in the nearest neighbor method. *Pattern Recognit. Image Anal.*, 20(4):484–494, December 2010.

Support Vector Machines

Armin Shmilovici

1 Introduction

Support vector machines (SVMs) are a set of related methods for supervised learning, applicable to both classification and regression problems. Since the introduction of the SVM classifier over two decades ago (Vapnik 1995), SVM gained popularity due to its solid theoretical foundation. The development of efficient implementations led to numerous applications (Nayak et al. 2015).

The support vector learning machine was developed by Vapnik et al. (Scholkopf et al. 1995; Scholkopf 1997) to constructively implement principles from *statistical learning* theory (Vapnik 2013). In the statistical learning framework, learning means to estimate a function from a set of examples (the training sets). To do this, a learning machine must choose one function from a given set of functions, which minimizes a certain risk (the *empirical risk*) that the estimated function is different from the actual (yet unknown) function. However, the risk depends on the complexity of the set of functions chosen as well as on the training set. Thus, a learning machine must find the best set of functions – as determined by its complexity –and the best function in that set. Unfortunately, in practice, a bound on the risk is neither easily computable, nor very helpful for analyzing the quality of the solution (Vapnik and Chapelle 2000).

Let us assume, for the moment, that the training set is separable by a *hyperplane*. It has been proved (Vapnik 1998) that for the class of hyperplanes, the complexity of the hyperplane can be bounded in terms of another quantity, the *margin*. The margin is defined as the minimal distance of an example to a decision surface. Thus, if we bound the margin of a function class from below, we can control its complexity.

A. Shmilovici (✉)
Ben-Gurion University, Beersheba, Israel
e-mail: armin@bgu.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_6

Support vector learning implements this insight that the *risk is minimized when the margin is maximized*. An SVM chooses a maximum-margin hyperplane that lies in a transformed input space and splits the example classes while maximizing the distance to the nearest cleanly split examples. The parameters of the solution hyperplane are derived from a quadratic programming optimization problem.

For example, consider a simple separable classification method in multi-dimensional space. Given two classes of examples clustered in feature space, any reasonable classifier hyperplane should pass *between* the means of the classes. *One* possible hyperplane is the decision surface that assigns a new point to the class whose mean is *closer* to it. This decision surface is geometrically equivalent to computing the class of a new point by checking the angle between two vectors – the vector connecting the two cluster means and the vector connecting the mid-point on that line with the new point. This angle can be formulated in terms of *a dot product* operation between vectors. The decision surface is implicitly defined in terms of the *similarity* between any new point and the cluster mean –- *a kernel function*. This simple classifier is linear in the feature space while in the input domain it is represented by a kernel expansion in terms of the training examples. In the more sophisticated techniques presented in the next section, the selection of the examples that the kernels are centered on will no longer consider *all* training examples, and the weights that are put on each data point for the decision surface will no longer be uniform. For instance, we might want to remove the influence of examples that are far away from the decision boundary, either since we expect that they will not improve the generalization error of the decision function, or since we would like to reduce the computational cost of evaluating the decision function. Thus, the hyperplane will only depend on a subset of training examples, called *support vectors*.

There are numerous books and tutorial papers on the theory and practice of SVM (Scholkopf and Smola 2002; Cristianini and Shawe-Taylor 2000; Muller et al. 2001; Chen et al. 2003; Smola and Scholkopf 2004; Deng et al. 2012). The aim of this chapter is to introduce the main SVM models and discuss their main attributes in the framework of supervised learning. The rest of this chapter is organized as follows: Sect. 2 describes the separable classifier case and the concept of kernels; Sect. 3 presents the non-separable case and some related SVM formulations; Sect. 4 discusses some practical computational aspects; Sect. 5 discusses some related concepts and applications; and Sect. 6 concludes with a discussion.

2 Hyperplane Classifiers

The task of classification is to find a rule, which based on external observations, assigns an object to one of several classes. In the simplest case, there are only two different classes. One possible formalization of this classification task is to estimate a function $f : \mathbf{R}^N \rightarrow \{-1, +1\}$ using input-output training data pairs generated identically and independently distributed (i.i.d.) according to an unknown

probability distribution $P(\mathbf{x}, y)$ of the data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbf{R}^N \times Y$, $Y = \{-1, +1\}$ such that f will correctly classify unseen examples $(\mathbf{x}, y)$. The test examples are assumed to be generated from the same probability distribution as the training data. An example is assigned to class $+1$ if $f(x) \geq 0$ and to class -1 otherwise.

The best function f that one can obtain is the one minimizing the expected error (risk) – the integral of a certain *loss function* l according to the unknown probability distribution $P(\mathbf{x}, y)$ of the data. For classification problems, l is the so-called 0/1 loss function: $l(f(\mathbf{x}), y) = \theta(-yf(\mathbf{x}))$, where $\theta(z) = 0$ for $z < 0$ and $\theta(z) = 1$ otherwise. The loss framework can also be applied to regression problems where $y \in \mathbf{R}$, where the most common loss function is the squared loss: $l(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$.

Unfortunately, the risk cannot be minimized directly, since the underlying probability distribution $P(\mathbf{x}, y)$ is unknown. Therefore, we must try to estimate a function that is close to the optimal one based on the available information, i.e., the training sample and properties of the function class from which the solution f is chosen. To design a learning algorithm, one needs to come up with a class of functions whose capacity (to classify data) can be computed. The intuition, which is formalized in Vapnik (2013), is that a simple (e.g., linear) function that explains most of the data is preferable to a complex one (Occam's razor).

2.1 The Linear Classifier

Let us assume for a moment that the training sample is separable by a hyperplane (see Fig. 1) and we choose functions of the form

$$(\mathbf{w} \cdot \mathbf{x}) + b = 0 \quad \mathbf{w} \in \mathbf{R}^N, b \in \mathbf{R} \tag{1}$$

corresponding to decision functions

$$f(\mathbf{x}) = \operatorname{sign}((\mathbf{w} \cdot \mathbf{x}) + b) \tag{2}$$

It has been shown (Vapnik 1995) that, for the class of hyperplanes, the capacity of the function can be bounded in terms of another quantity, the margin (Fig. 1). The margin is defined as the minimal distance of a sample to the decision surface. The margin depends on the length of the weight vector $\mathbf{w}$ in Eq. (1): since we assumed that the training sample is separable, we can rescale $\mathbf{w}$ and b such that the points closest to the hyperplane satisfy $|(\mathbf{w} \cdot \mathbf{x}_i) + b| = 1$ (i.e., obtain the so-called canonical representation of the hyperplane). Now consider two samples $\mathbf{x}_1$ and $\mathbf{x}_2$ from different classes with $|(\mathbf{w} \cdot \mathbf{x}_1) + b| = 1$ and $|(\mathbf{w} \cdot \mathbf{x}_2) + b| = 1$, respectively. Then, the margin is given by the distance of these two points, measured perpendicular to the hyperplane, i.e., $\left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}_1 - \mathbf{x}_2)\right) = \frac{2}{\|\mathbf{w}\|}$.

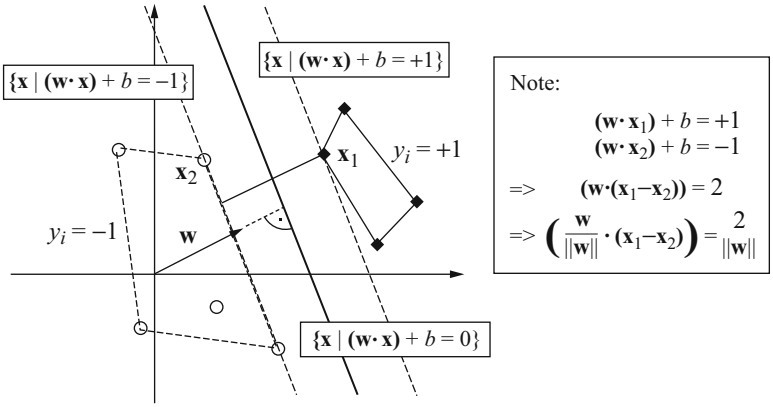


Fig. 1 A toy binary classification problem: separate balls from diamonds. The optimal hyperplane is orthogonal to the shortest line connecting the convex hull of the two classes (dotted), and intersects it half way between the two classes. In this case, the margin is measured perpendicular to the hyperplane. (Figure taken from Muller et al. (2001))

Among all the hyperplanes separating the data, there exists a unique one yielding the maximum margin of separation between the classes:

$$\underset{\{\mathbf{W},b\}}{\text{Max}}\ \min\left\{\|\mathbf{x}-\mathbf{x}_i\| : \mathbf{x}\in\mathbf{R}^N, \left(\mathbf{w}\cdot\mathbf{x}\right)\right) + b = 0, i = 1,\ldots n\right\} \tag{3}$$

To construct this optimal hyperplane, one solves the following optimization problem:

$$\underset{\{\mathbf{W},b\}}{\text{Min}}\ \frac{1}{2}\|\mathbf{w}\|^2 \tag{4}$$

$$\text{Subject to } y_i\cdot((\mathbf{w}\cdot\mathbf{x}_i)+b)\geq 1,\ i=1,\ldots,n \tag{5}$$

This constraint optimization problem can be solved by introducing Lagrange multipliers $\alpha_i \geq 0$ and the Lagrangian function

$$L\left(\mathbf{w},b,\boldsymbol{\alpha}\right) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n}\alpha_i\left(y_i\cdot((\mathbf{w}\cdot\mathbf{x}_i)+b)-1\right) \tag{6}$$

The Lagrangian L has to be minimized with respect to the *primal* variables $\{\mathbf{w}, b\}$ and maximized with respect to the *dual* variables α_i. The optimal point is a saddle point and we have the following equations for the primal variables:

$$\frac{\partial L}{\partial b} = 0,\ \frac{\partial L}{\partial \mathbf{W}} = 0 \tag{7}$$

which translate into

$$\sum_{i=1}^{n} \alpha_i y_i = 0\,,\ \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{8}$$

The solution vector thus has an expansion in terms of a subset of the training patterns. The *support vectors* are those patterns corresponding with the non-zero α_i, and the non-zero α_i are called *support values*. By the Karush-Kuhn-Tucker complimentary conditions of optimization, the α_i must be zero for all the constraints in (5) which are not met as equality, thus

$$\alpha_i \left(y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1\right) = 0\,,\ i = 1, \ldots, n \tag{9}$$

and all the support vectors lie on the margin (Figs. 1 and 3) while all the remaining training examples are irrelevant to the solution. The hyperplane is completely captured by the patterns closest to it.

For a nonlinear problem like (4) and (5), called a primal problem, under certain conditions, the primal and dual problems have the same objective values. Therefore, we can solve the dual problem which may be easier than the primal problem. In particular, when working in feature space (Sect. 2.3) solving the dual may be the only way to train the SVM. By substituting (8) into (6), one eliminates the primal variables and arrives at the Wolfe dual (Wolfe 1961) of the optimization problem for the multipliers α_i:

$$\underset{\boldsymbol{\alpha}}{\text{Max}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \tag{10}$$

$$\text{Subject to } \alpha_i \geq 0,\ \ i = 1, \ldots, n,\ \ \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{11}$$

The hyperplane decision function (2) can now be explicitly written as

$$f(\mathbf{x}) = \ \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \left(\mathbf{x} \cdot \mathbf{x}_i\right) + b\right) \tag{12}$$

where b is computed from (9) and from the set of support vectors $\mathbf{x}_i$, $i \in I \equiv \{i : \alpha_i \neq 0\}$.

$$b = \frac{1}{|I|} \sum_{i \in I} \left(y_i - \sum_{j=1}^{n} \alpha_j y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)\right) \tag{13}$$

2.2 The Kernel Trick

The choice of linear classifier functions seems to be very limited (i.e., likely to underfit the data). Fortunately, it is possible to have both linear models and a very rich set of nonlinear decision functions by using the kernel trick (Cortes and Vapnik 1995) with maximum-margin hyperplanes. Using the kernel trick for SVM makes the maximum-margin hyperplane be fit in a feature space F. The feature space F is a nonlinear map $\Phi : R^N \rightarrow F$ from the original input space, usually of much higher dimensionality than the original input space. With the kernel trick, the same linear algorithm is worked on the transformed data $(\Phi(\mathbf{x}_1), y_1), \ldots, ((\Phi(\mathbf{x}_n), y_n))$. In this way, nonlinear SVMs can make the maximum-margin hyperplane fit in a feature space. Figure 2 demonstrates such a case. In the original (linear) training algorithm (10), (11), and (12) the data appears in the form of dot products $\mathbf{x}_i \cdot \mathbf{x}_j$. Now, the training algorithm depends on the data through dot products in F, i.e., on functions of the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. If there exists a kernel function K such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, we would only need to use K in the training algorithm and would never need to explicitly even know what Φ is.

Mercer's condition (Vapnik 2013) tells us the mathematical properties to check whether or not a prospective kernel is actually a dot product in some space, but it does not tell us how to construct Φ, or even what F is. Choosing the best kernel function is a subject of active research (Scholkopf and Smola 2002; Steinwart 2003). It was found that to a certain degree different choices of kernels give similar classification accuracy and similar sets of support vectors (Scholkopf et al.

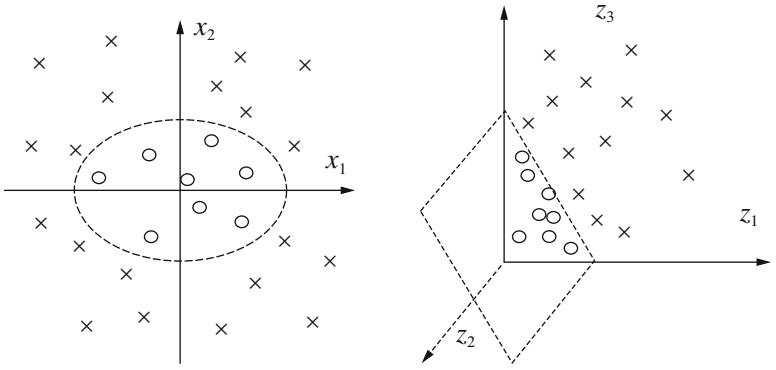


Fig. 2 The idea of SVM is to map the training data into a higher dimensional feature space via Φ, and construct a separating hyperplane with maximum margin there. This yields a nonlinear decision boundary in input space. In the following two-dimensional classification example, the transformation is $\Phi : \mathbf{R}^2 \rightarrow \mathbf{R}^3$, $(x_1, x_2) \rightarrow (z_1, z_2, z_3) \equiv \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$. The separating hyperplane is visible and the decision surface can be analytically found. (Figure taken from Muller et al. (2001))

Table 1 Commonly used kernel functions

Kernel	$K(\mathbf{x}, \mathbf{x}_i)$
Radial Basis Function	$\exp\left(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\right),\ \gamma > 0$
Inverse multiquadratic	$\frac{1}{\sqrt{\|\mathbf{x}-\mathbf{x}_i\|+\eta}}$
Polynomial of degree d	$\left(\gamma\left(\mathbf{x}^T \cdot \mathbf{x}_i\right) + \eta\right)^d,\ \gamma > 0$
Sigmoidal	$\tanh\left(\gamma\left(\mathbf{x}^T \cdot \mathbf{x}_i\right) + \eta\right),\ \gamma > 0$
Linear	$\mathbf{x}^T \cdot \mathbf{x}_i$

1995), indicating that in some sense there exist "important" training points which characterize a given problem.

Some commonly used kernels are presented in Table 1. Note, however, that the Sigmoidal kernel only satisfies the Mercer's condition for certain values of the parameters and the data. Hsu et al. (2016) advocated the use of the Radial Basis Function as a reasonable first choice.

2.3 The Optimal Margin Support Vector Machine

Using the kernel trick, replace every dot product $(\mathbf{x}_i \cdot \mathbf{x}_j)$ in terms of the kernel K evaluated on input patterns $\mathbf{x}_i, \mathbf{x}_j$. Thus, we obtain the more general form of (12):

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{14}$$

and the following quadratic optimization problem

$$\underset{\boldsymbol{\alpha}}{\text{Max}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) \tag{15}$$

$$\text{Subject to } \alpha_i \geq 0,\ \ i = 1, \ldots, n,\ \ \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{16}$$

Formulation (15) and (16) is the standard SVM formulation. This dual problem has the same number of variables as the number of training variables, while the primal problem has a number of variables which depends on the dimensionality of the feature space, which could be infinite. Figure 3 presents an example of a decision function found with an SVM.

One of the most important properties of the SVM is that the solution is *sparse* in $\boldsymbol{\alpha}$, i.e., many patterns are outside the margin area and their optimal $\boldsymbol{\alpha}_i$ is zero. Without this sparsity property, SVM learning would hardly be practical for large data sets.

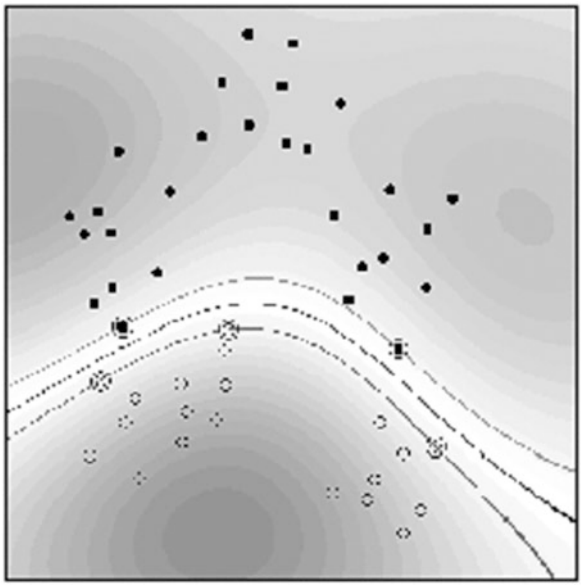


Fig. 3 Example of a support vector classifier found by using a radial basis function kernel. Circles and disks are two classes of training examples. Extra circles mark the support vectors found by the algorithm. The middle line is the decision surface. The outer lines precisely meet the constraint (16). The shades indicate the absolute value of the argument of the sign function in (14). (Figure taken from Chen et al. (2003))

3 Non-separable SVM Models

The previous section considered the separable case. However, in practice, a separating hyperplane may not exist, e.g., if a high noise level causes some overlap of the classes. Using the previous SVM might not minimize the empirical risk. This section presents some SVM models that extend the capabilities of hyperplane classifiers to more practical problems.

3.1 Soft Margin Support Vector Classifiers

To allow for the possibility of examples violating constraint (5), Cortes and Vapnik (1995) introduced slack variables ξ_i that relax the hard margin constraints

$$y_i \cdot ((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b) \geq 1 - \xi_i\,,\ \ \xi_i \geq 0,\ i = 1, \ldots, n \tag{17}$$

A classifier that generalizes well is then found by controlling both the classifier capacity (via $\|\mathbf{w}\|$) and the sum of the slacks $\sum_{i=1}^{n} \xi_i$, i.e., the number of training errors. One possible realization, called C-SVM, of a soft margin classifier is minimizing the following objective function

$$\underset{\mathbf{w},b,\xi}{\text{Minimize}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{18}$$

The regularization constant $C > 0$ determines the trade-off between the empirical error and the complexity term. Incorporating Lagrange multipliers and solving leads to the following dual problem:

$$\underset{\boldsymbol{\alpha}}{\text{Max}} \ \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) \tag{19}$$

$$\text{Subject to } 0 \le \alpha_i \le C, \ \ i = 1, \ldots, n, \ \ \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{20}$$

The only difference from the separable case is the upper bound C on the Lagrange multipliers α_i. The solution remains sparse and the decision function retains the same form as (14).

Another possible realization of a soft margin, called ν-SVM (Chen et al. 2003), was originally proposed for regression. The rather non-intuitive regularization constant C is replaced with another constant $\nu \in [0, 1]$. The dual formulation of the ν-SVM is the following:

$$\underset{\boldsymbol{\alpha}}{\text{Max}} \ -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) \tag{21}$$

$$\text{Subject to } 0 \le \alpha_i \le \tfrac{1}{n}, \ \ i = 1, \ldots, n, \ \ \sum_{i=1}^{n}\alpha_i y_i = 0, \ \ \sum_{i=1}^{n}\alpha_i \ge \nu \tag{22}$$

For appropriate parameter choices, the ν-SVM yields exactly the same solutions as the C-SVM. The significance of ν is that under some mild assumptions about the data, ν is an upper bound on the fraction of margin errors (and hence also on the fraction of training errors); and ν is also a lower bound on the fraction of support vectors. Thus, controlling ν influences the trade-off between the model's accuracy and the model's complexity.

3.2 Support Vector Regression

One possible formalization of the regression task is to estimate a function $f : \mathbf{R}^N \to \mathbf{R}$ using input-output training data pairs generated identically and independently distributed (i.i.d.) according to an unknown probability distribution $P(\mathbf{x}, y)$ of the data. The concept of margin is specific to classification. However, we

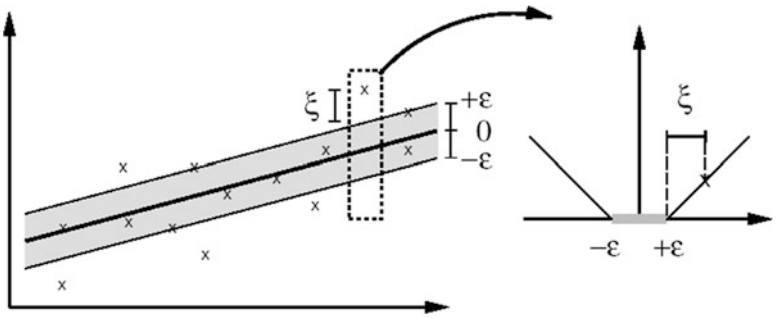


Fig. 4 In SV regression, a tube with radius ε is fitted to the data. The optimization determines a trade-off between model complexity and points lying outside of the tube. (Figure taken from Smola and Scholkopf (2004))

would still like to avoid too complex regression functions. The idea of SVR (Smola and Scholkopf 2004) is that we find a function that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible. In other words, errors are unimportant as long as they are less than ε, but we do not tolerate deviations larger than this. An analog of the margin is constructed in the space of the target values $y \in \mathbf{R}$. By using Vapnik's ε-sensitive loss function (Fig. 4).

$$|y - f(\mathbf{x})|_\varepsilon \equiv \max\{0, |y - f(\mathbf{x})| - \varepsilon\} \tag{23}$$

A tube with radius ε is fitted to the data, and a regression function that generalizes well is then found by controlling both the regression capacity (via $\|\mathbf{w}\|$) and the loss function. One possible realization, called C-SVR, of a is minimizing the following objective function

$$\underset{\mathbf{w},b,\xi}{\text{Minimize}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} |y_i - f(\mathbf{x})|_\varepsilon \tag{24}$$

The regularization constant $C > 0$ determines the trade-off between the empirical error and the complexity term.

Generalization to kernel-based regression estimation is carried out in complete analogy with the classification problem. Introducing Lagrange multipliers and choosing a priori the regularization constants C, ε one arrives at a dual quadratic optimization problem. The support vectors and the support values of the solution define the following regression function

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{25}$$

There are degrees of freedom for constructing SVR, such as how to penalize or regularize different parts of the vector, how to use the kernel trick, and the loss function to use. For example, in the ν-SVR algorithm implemented in LIBSVM (Chang and Lin 2011) one specifies an upper bound $0 \leq \nu \leq 1$ on the fraction of points allowed to be outside the tube (asymptotically, the number of support vectors). For a priori chosen constants C, ν the dual quadratic optimization problem is as follows:

$$\underset{\boldsymbol{\alpha},\boldsymbol{\alpha}^*}{\text{Max}} \sum_{i=1}^{n} \left(\alpha_i^* - \alpha_i\right) y_i - \frac{1}{2} \sum_{i,j=1}^{n} \left(\alpha_i^* - \alpha_i\right) \left(\alpha_j^* - \alpha_j\right) K\left(\mathbf{x}_i, \mathbf{x}_j\right) \tag{26}$$

$$\text{Subject to } 0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{n}, \begin{array}{l} \sum_{i=1}^{n} \left(\alpha_i^* + \alpha_i\right) \leq C\nu \\ \sum_{i=1}^{n} \left(\alpha_i^* - \alpha_i\right) \leq C\nu \end{array} \quad i = 1, \ldots, n \tag{27}$$

and the regression solution is expressed as

$$f\left(\mathbf{x}\right) = \sum_{i=1}^{n} \left(\alpha_i^* - \alpha_i\right) K\left(\mathbf{x}, \mathbf{x}_i\right) + b \tag{28}$$

3.3 SVM-Like Models

The power of SVM comes from the kernel representation that allows a nonlinear mapping of input space to a higher dimensional feature space. However, the resulting quadratic programming equations may be computationally expensive for large problems. Smola et al. (1999) suggested an SVR-like *linear programming* formulation that retains the form of the solution (25) while replacing the quadratic function (26) with a linear function subject to constraints on the error of kernel expansion (25).

Suykens et al. (2002) introduced the least squares SVM (LS-SVM) in which they modify the classifier of Eqs. (17) and (18) with the following equations:

$$\underset{\mathbf{w},b,\mathbf{e}}{\text{Minimize}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{i=1}^{n} e_i^2 \tag{29}$$

$$\text{Subject to } \ y_i \cdot \left(\left(\mathbf{w} \cdot \Phi\left(\mathbf{x}_i\right)\right) + b\right) = 1 - e_i, \quad i = 1, \ldots, n \tag{30}$$

Important differences with standard SVM are the equality constraint (30) and the sum squared error terms, which greatly simplify the problem. Incorporating Lagrange multipliers and solving leads to the following dual *linear problem*:

$$\begin{bmatrix} \mathbf{0} & \mathbf{Y}^T \\ \mathbf{Y} & \boldsymbol{\Omega} + \gamma^{-1}\mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \tag{31}$$

where the primal variables $\{\mathbf{w}, b\}$ define as before a decision surface like (14), $Y = (y_1, \ldots, y_n)$, $(\Omega)_{i,j} = y_i y_j K(x_i, x_j)$, $\mathbf{I}$,$\mathbf{0}$ are appropriate size all ones (all zeros) matrices, and γ is a tuning parameter to be optimized. Equivalently, modifying the regression problem (26) and (27) also results in a linear system like (31) with additional tuning parameter.

The LS-SVM can realize strongly nonlinear decision boundaries, and efficient matrix inversion methods can handle very large datasets. However, $\boldsymbol{\alpha}$ is *not* sparse anymore (Suykens et al. 2002).

4 Implementation Issues with SVM

The purpose of this section is to overview some problems that face the application of SVM in machine learning.

4.1 Optimization Techniques

The solution of the SVM problem is the solution of a constraint (convex) quadratic programming (QP) problem such as (15) and (16). Equation (15) can be rewritten as maximizing $-\frac{1}{2}\boldsymbol{\alpha}^T \hat{\mathbf{K}} \boldsymbol{\alpha} + \mathbf{1}^T \boldsymbol{\alpha}$, where $\mathbf{1}$ is a vector of all ones and $\hat{K}_{i,j} = y_i y_j k\left(x_i, x_j\right)$. When the Hessian matrix $\hat{\mathbf{K}}$ is positive definite, the objective function is convex and there is a *unique global solution*. If matrix $\hat{\mathbf{K}}$ is positive semi-definite, every maximum is also a global maximum, however, there can be several optimal solutions (different in their $\boldsymbol{\alpha}$) which might lead to different performance on the testing dataset.

In general, the support vector optimization can be solved analytically only when the number of training data is very small. The worst case computational complexity for the general analytic case results from the inversion of the Hessian matrix, thus is of order N_S^3, where N_S is the number of support vectors. There exists a vast literature on solving quadratic programs (Bertsekas 2016; Bazaraa et al. 2006) and several software packages are available. However, most quadratic programming algorithms are either only suitable for small problems or assume that the Hessian matrix $\hat{\mathbf{K}}$ is sparse, i.e., most elements of this matrix are zero. Unfortunately, this is not true for the SVM problem. Thus, using standard quadratic programming codes with more than a few hundred variables results in enormous training times and more demanding memory needs. Nevertheless, the structure of the SVM optimization problem allows the derivation of specially tailored algorithms, which allow for fast convergence with small memory requirements, even on large problems.

A key observation in solving large-scale SVM problems is the sparsity of the solution (Steinwart 2004). Depending on the problem, many of the optimal α_i will either be zero or on the upper bound. If one could know beforehand which α_i were zero, the corresponding rows and columns could be removed from the matrix $\hat{\mathbf{K}}$ without changing the value of the quadratic form. Furthermore, a point can only be optimal if it fulfills the KKT conditions (such as Eq. (5)). SVM solvers decompose the quadratic optimization problem into a sequence of smaller quadratic optimization problems that are solved in sequence. Decomposition methods are based on the observations of Osuna et al. (1997) that each QP in a sequence of QPs always contains at least one sample violating the KKT conditions. The classifier built from solving the QP for part of the training data is used to test the rest of the training data. The next partial training set is generated from combining the support vectors already found (the "working set") with the points that most violate the KKT conditions, such that the partial Hessian matrix will fit the memory. The algorithm will eventually converge to the optimal solution. Decomposition methods differ in the strategies for generating the smaller problems and use sophisticated heuristics to select several patterns to add and remove from the sub-problem plus efficient caching methods. They usually achieve fast convergence even on large data sets with up to several thousands of support vectors. A quadratic optimizer is still required as part of the solver. Elements of the SVM solver can take advantage of parallel processing: such as simultaneous computing of the Hessian matrix, dot products, and the objective function. More details and tricks can be found in the literature (Cai and Cherkassky 2012; Tavara 2019; Smola et al. 2000; Deng et al. 2012; Chang and Lin 2001; Chew et al. 2003; Chung et al. 2004). For real-time pattern recognition, on-line SVM algorithms (Zhou et al. 2015) avoid full SVM retraining when comes a new sample via off-line pre-selection of few important training data (Wang 2013).

A fairly large selection of optimization codes for SVM classification and regression has been developed. They range from simple MATLAB implementation to sophisticated C, C++, or FORTRAN programs (e.g., LIBSVM: Chang and Lin 2011, SVMlight: Joachims 2008). Some solvers include integrated model selection and data rescaling procedures for improved speed and numerical stability. Hsu et al. (2016) advise about working with an SVM software on practical problems.

4.2 Model Selection

To obtain a high level of performance, some parameters of the SVM algorithm have to be tuned. These include (1) the selection of the kernel function; (2) the kernel parameter(s); (3) the regularization parameters (C, ν, ε) for the trade-off between the model complexity and the model accuracy. Model selection techniques provide principled ways to select a proper kernel. Usually, a sequence of models is solved, and using some heuristic rules, next set of parameters is tested. The process is continued until a given criterion is obtained (e.g., 99% correct classification). For example, if we consider 3 alternative (single parameter) kernels, 5 partitions of the

kernel parameters, and one regularization parameter with 5 partitions each, then we need to consider a total of $3 \times 5 \times 5 = 125$ SVM evaluations.

The *cross-validation* technique is widely used for the prediction of the generalization error, and is included in some SVM packages (such as LIBSVM: Chang and Lin 2011). Here, the training samples are divided into k subsets of equal size. Then, the classifier is trained k times: in the i-th iteration ($i = 1,...,k$), the classifier is trained on all subsets except the i-th one. Then, the classification error is computed for the i-th subset. It is known that the average of these k errors is a rather good estimate of the generalization error. k is typically 5 or 10. Thus, for the example above we need to consider at least 625 SVM evaluations to identify the model of the best SVM classifier.

In the *Bayesian evidence framework* the training of an SVM is interpreted as Bayesian inference, and the model selection is accomplished by maximizing the marginal likelihood (i.e., evidence). Law and Kwok (2000) and Wenzel et al. (2017) provide iterative parameter updating formulas, and report a significantly smaller number of SVM evaluations.

4.3 Multi-Class SVM

Though SVM was originally designed for two-class problems, several approaches have been developed to extend SVM for multi-class data sets.

One approach to k-class pattern recognition is to consider the problem as a collection of binary classification problems. The technique of *one-against-the-rest* requires k binary classifiers to be constructed (when the label $+1$ is assigned to each class in its turn and the label -1 is assigned to the other $k - 1$ classes). In the prediction stage, a voting scheme is applied to classify a new point. In the *winner-takes-all* voting scheme, one assigns the class with the largest real value. The *one-against-one* approach trains a binary SVM for any two classes of data and obtains a decision function. Thus, for a k-class problem, there are $k(k-1)/2$ decision functions where the voting scheme is designated to choose the class with the maximum number of votes. More elaborate voting schemes, such as *error-correcting-codes,* consider the combined outputs from the n-parallel classifiers as a binary n-bit code word and select the class with the closest (e.g., Hamming distance) code.

In Hsu and Lin (2002), it was experimentally shown that for general problems, using the C-SVM classifier, various multi-class approaches give similar accuracy. Rifkin and Klautau (2004) have similar observation, however, this may not always be the case. Multi-class methods must be considered together with parameter-selection strategies. That is, we search for appropriate regularization parameters and kernel parameters for constructing a better model (Didiot and Lauer 2015). Chen, Lin and Scholkopf (2003) experimentally demonstrate inconsistent and marginal improvement in the accuracy when the parameters are trained differently for each classifier inside a multi-class C-SVM and ν-SVM classifiers.

5 Extensions and Application

Kernel algorithms have solid foundations in statistical learning theory and functional analysis; thus, kernel methods combine statistics and geometry. Kernels provide an elegant framework for studying fundamental issues of machine learning, such as similarity measures that can incorporate prior knowledge about the problem, and data representations. SVM have been one of the major kernel methods for supervised learning. It is not surprising that recent methods integrate SVM with kernel methods (Scholkopf et al. 1999; Scholkopf and Smola 2002; Shawe-Taylor and Cristianini 2004) for unsupervised learning problems such as density estimation (Weston and Herbrich 2000).

SVM has a strong analogy in *regularization theory* (Williamson et al. 2001). Regularization is a method of solving problems by making some a priori assumptions about the desired function. A penalty term that discourages over-fitting is added to the error function. A common choice of regularizer is given by the sum of the squares of the weight parameters and results in a functional similar to (6). Like SVM, optimizing a functional of the learning function, such as its smoothness, leads to sparse solutions.

Boosting is a machine learning technique that attempts to improve a "weak" learning algorithm, by a convex combination of the original "weak" learning function, each one trained with a different distribution of the data in the training set. SVM can be translated to a corresponding boosting algorithm using the appropriate regularization norm (Ratsch et al. 2001).

Successful applications of SVM algorithms have been reported for various fields, such as pattern recognition (Martin et al. 2002), text categorization (Dumais 1998; Joachims 2002), time series prediction (Mukherjee et al. 1997), and bio-informatics (Zien et al. 2000). Historically, classification experiments with the U.S. Postal Service benchmark problem – the first real-world experiment of SVM (Cortes and Vapnik 1995; Scholkopf 1997) – demonstrated that plain SVMs give a performance very similar to other state-of-the-art methods. SVMs have been achieving excellent results also on the Reuters-22173 text classification benchmark problem (Dumais 1998). SVMs have been strongly improved by using prior knowledge about the problem to engineer the kernels and the support vectors with techniques such as virtual support vectors (Scholkopf 1997; Scholkopf et al. 1998). Nayak et al. (2015) present many more applications.

6 Conclusion

Since the introduction of the SVM classifier over two decades ago, SVM gained popularity due to its solid theoretical foundation in statistical learning theory. They differ radically from comparable approaches such as neural networks: they have a simple geometrical interpretation and SVM training always finds a global mini-

mum. The development of efficient implementations led to numerous applications. Selected real-world applications served to exemplify that SVM learning algorithms are indeed highly competitive on a variety of problems.

SVMs are a set of related methods for supervised learning, applicable to both classification and regression problems. This chapter provides an overview of the main SVM methods for the separable and non-separable case and for classification and regression problems. However, SVM methods are being extended to unsupervised learning problems.

An SVM is largely characterized by the choice of its kernel. The kernel can be viewed as a nonlinear similarity measure, and should ideally incorporate prior knowledge about the problem at hand. The best choice of kernel for a given problem is still an open research issue. A second limitation is the speed of training. Training for very large datasets (millions of support vectors) is still an unsolved problem.

References

Bazaraa M. S., Sherali H. D., and Shetty C. M. Nonlinear programming: theory and algorithms. Wiley, third edition, 2006.

Bertsekas D.P. Nonlinear Programming. Athena Scientific, third edition, 2016.

Cai, F., Cherkassky, V.: Generalized SMO algorithm for SVM-based multitask learning. IEEE Trans. Neural Network Learning Systems 2012; **23**(6):997–1003.

Chang C.-C. and Lin C.-J. Training support vector classifiers: Theory and algorithms. Neural Computation 2001; 13(9):2119–2147.

Chang C.-C. and Lin C.-J. LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology. 2011; 2:(27):1–27. Software available at https://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Chen P.-H., Lin C. -J., and Scholkopf B. A tutorial on nu-support vector machines. 2003.

Chew H. G., Lim C. C., and Bogner R. E. An implementation of training dual-nu support vector machines. In Qi, Teo, and Yang, editors, Optimization and Control with Applications. Kluwer, 2003.

Chung K.-M., Kao W.-C., Sun C.-L., and Lin C.-J. Decomposition methods for linear support vector machines. Neural Computation 2004; 16(8):1689–1704.

Cortes C. and Vapnik V. Support vector networks. Machine Learning 1995; 20:273–297.

Cristianini N. and Shawe-Taylor J. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

Deng N., Tian Y., Zhang C. Support Vector Machines Optimization Based Theory, Algorithms, and Extensions. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, 2012.

Didiot E. and Lauer F., Efficient optimization of multi-class support vector machines with MSVMpack. In Modelling, Computation and Optimization in Information Systems and Management Sciences. Springer, 23–34, 2015.

Dumais S. Using SVMs for text categorization. IEEE Intelligent Systems 1998; 13(4).

Hsu C.-W. and Lin C.-J. A comparison of methods for multi-class support vector machines IEEE Transactions on Neural Networks 2002; 13(2); 415–425.

Hsu C.-W. Chang C.-C and Lin C.-J. A practical guide to support vector classification. 2016. Available Online: www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Joachims T. Learning to Classify Text using Support Vector Machines Methods, Theory, and Algorithms. Kluwer Academic Publishers, 2002.

Joachims T. 2008, SVMlight, available online http://www.cs.cornell.edu/People/tj/svm_light/

Law M. H. and Kwok J. T. Bayesian support vector regression. Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics (AISTATS) pages 239–244, Key-West, Florida, USA, January 2000.

Martin D. R., Fowlkes C. C., and Malik J. Learning to detect natural image boundaries using brightness and texture. In Advances in Neural Information Processing Systems, volume 14, 2002.

Mukherjee S., Osuna E., and Girosi F. Nonlinear prediction of chaotic time series using a support vector machine. In Principe J., Gile L., Morgan N. and Wilson E. editors, Neural Networks for Signal Processing VII - proceedings of the 1997 IEEE Workshop, pages 511–520, New-York, IEEE Press, 1997.

Muller K.-R., Mika S., Ratsch G., Tsuda K., and Scholkopf B., An introduction to kernel-based learning algorithms. *IEEE Neural Networks* 2001; 12(2):181–201.

Nayak J, Naik B, and Behera H. S. A comprehensive survey on support vector machine in data mining tasks: Applications & challenges. International Journal of Database Theory and Application 2015; 8,(1):169–186.

Osuna E., Freund R., and Girosi F. An improved training algorithm for support vector machines. In Principe J., Gile L., Morgan N. and Wilson E. editors, Neural Networks for Signal Processing VII - proceedings of the 1997 IEEE Workshop, pages 276–285, New-York, IEEE Press, 1997.

Ratsch G., Onoda T., and Muller K.R. Soft margins for AdaBoost. Machine Learning 2001; 42(3):287–320.

Rifkin R. and Klautau A.. In Defense of One-vs-All Classification, Journal of Machine Learning Research 2004; 5:101–141.

Scholkopf B., Support Vector Learning. Oldenbourg Verlag, Munich, 1997.

Scholkopf B., Burges C.J.C., and Vapnik V.N. Extracting support data for a given task. In Fayyad U.M. and Uthurusamy R., Editors, Proceedings, First International Conference on Knowledge Discovery and Data Mining. AAAI Press, Menlo Park, CA, 1995.

Scholkopf B., Simard P.Y., Smola A.J., and Vapnik V.N.. Prior knowledge in support vector kernels. In Jordan M., Kearns M., and Solla S., Editors, Advances in Neural Information Processing Systems 10, pages 640–646. MIT Press, Cambridge, MA, 1998.

Scholkopf B., Burges C. J. C., and Smola A. J., editors, Advances in Kernel Methods - Support Vector Learning, Cambridge, MA, MIT Press, 1999.

Scholkopf B. and Smola A. J. Learning with Kernels. MIT Press, Cambridge, MA, 2002.

Scholkopf B., Smola A. J., Williamson R. C., and Bartlett P. L. New support vector algorithms. Neural Computation 1999; 12:1207–1245.

Shawe-Taylor J. and Cristianini N. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

Smola A. J., Bartlett P. L., Scholkopf B. and Schuurmans D. Advances in Large Margin Classifiers. MIT Press, Cambridge, MA, 2000.

Smola A.J. and Scholkopf B. A tutorial on support vector regression. Statistics and Computing 2004; 14(13):199–222.

Smola A.J., Scholkopf B. and Ratsch G. Linear programs for automatic accuracy control in regression. Proceedings of International Conference on Artificial Neural Networks ICANN'99, Berlin, Springer 1999.

Steinwart I. On the optimal parameter choice for nu-support vector machines. IEEE Transactions on Pattern Analysis and Machine Intelligence 2003; 23(10).

Steinwart I. Sparseness of support vector machines. Journal of Machine Learning Research 2004; 4(6):1071–1105.

Suykens J.A.K., Van Gestel T., De Brabanter J., De Moor B., and Vandewalle J. Least Squares Support Vector Machines. World Scientific Publishing , Singapore, 2002.

Tavara S. Parallel Computing of Support Vector Machines: A Survey. ACM Computing Surveys, 2019;. 51(6):1–38.

Vapnik V. The Nature of Statistical Learning Theory. Springer Verlag, New York, 1995.

Vapnik V. Statistical Learning Theory. Wiley, NY, 1998.

Vapnik V. The Nature of Statistical Learning Theory. Springer Science. 2013.

Vapnik V. and Chapelle O. Bounds on error expectation for support vector machines. Neural Computation 2000; 12(9):2013–2036.

Wang, D.I.: Online Support Vector Machine Based on Convex Hull Vertices Selection. IEEE Transactions on Neural Networks Learning Systems **2013; 24**(4), 593–608.

Weston J. and Herbrich R.. Adaptive margin support vector machines. In Smola A.J., Bartlett P.L., Scholkopf B., and Schuurmans D., Editors, Advances in Large Margin Classifiers, pages 281–296, MIT Press, Cambridge, MA, 2000,

Wenzel F. Galy-Fajou T. Deutsch M., Kloft M. Bayesian Nonlinear Support Vector Machines for Big Data. ECML/PKDD 2017: 307–322

Williamson R. C., Smola A. J., and Scholkopf B. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. IEEE Transactions on Information Theory 2001; 47(6):2516–2532.

Wolfe P. A duality theorem for non-linear programming. Quarterly of Applied Mathematics 1961; 19:239–244.

Zhou X., Zhang X. and Wang B., Online Support Vector Machine: A Survey, in Kim J.H and Geem Z. W. editors, Harmony Search Algorithm, pages 269–278, Proceedings of the 2nd International Conference on Harmony Search Algorithm (ICHSA 2015).

Zien A., Ratsch G., Mika S., Scholkopf B., Lengauer T. and Muller K.R. Engineering support vector machine kernels that recognize translation initiation sites. Bio-Informatics 2000; 16(9):799–807.

Empowering Interpretable, Explainable Machine Learning Using Bayesian Network Classifiers

Boaz Lerner

1 Introduction

From the seminal works of Pearl [1], Spirtes [2], Lauritzen and Spiegelhalter [3], Cooper [4], and Microsoft Research's researchers (mainly Heckerman [5, 6], Meek [7], and Chickering [8]) and their colleagues introducing Bayesian networks (BNs), to the works admitting and demonstrating BN classifiers (BNCs) [9, 10, 11, 12, 13, 14, 15, 16, 17, 18], BNs were mostly considered an esoteric field, a neglected little brother of the more popular mainstream neural networks, support vector machines, and boosting and bagging machines and their machine learning (ML) variants. BNs and BNCs have attracted the attention of only relatively few faithful, non-mainstream scientists in the ML community who recognized, cherished, and advanced the networks' huge potential.

Whether due to the NP-hard complexity of BN structure learning [8], the BN traditional near-exclusive focus on discrete data [4, 5, 6], or the frequency of new trends in ML, few in the ML community found study of the powerful BN theory and tools attractive. Traditionally presented as a knowledge representation paradigm, until the late 90s BNs were not considered accurate classifiers and works demonstrating their superb classification capability are unjustifiably scarce even today.

While in training a classifier, we have to minimize a loss function (usually the 0/1 loss function), which is equivalent to maximizing the classification accuracy of a single target variable, when learning a BN, we usually maximize/minimize a general likelihood-driven [4, 6] or similar-fitting function (Akaike information criterion (AIC) [19], Bayesian information criterion (BIC) [20], or Kullback–

B. Lerner (✉)
Ben-Gurion University of the Negev, Be'er Sheva, Israel
e-mail: boaz@bgu.ac.il
https://www.ee.bgu.ac.il/~boaz/

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_7

Leibler (KL) divergence [21]) over the set of all variables. However, as the domain size increases, maximization of a likelihood-driven function over many variables used in classification—of which only one is of real interest, i.e., the class variable—can hardly lead to an accurate classifier [9].

In contrast to the belief that BNCs are less accurate than traditional ML classifiers, several studies [11, 13, 17, 22] have shown that properly learned, BNCs are comparable to traditional ML classifiers.[1] In this chapter, we will present some of these studies. Beyond accuracy however, the use of BNCs should be considered for the natural interpretability (understanding the results; the "how" question) and explainability (explaining the results; the "why" question) they provide.[2]

Let us demonstrate the difficulty in achieving classifier interpretability using three examples. The linear regression model accompanies a predictor with a coefficient measuring "importance" and direction of impact, but with credibility that depends on the predictor value size. The ensemble classifier (whether by averaging methods such as bagging and the random forest (RF), or boosting methods such as the XGBoost) provides a ranked list of "important" variables but may show only slight differences between their levels of "importance"—say by a second or third digit after the decimal point—providing negligible information about their relative contribution to the classification. With hundreds and thousands of neurons residing in many internal modules and layers, connected by millions of parameters, the deep neural network (DNN) excels in classification but due to these complexities inherently lacks interpretability and explainability [30]. The BNC, in contrast to these examples, shows a hierarchy of interrelations among domain variables along with causal paths of influence and inference mechanisms, revealing causal relations that can readily be investigated while and for interpreting and explaining the domain.

ML tools are frequently accused of being black boxes,[3] sacrificing interpretability in favor of usability and effectiveness [31, 32, 33, 34]. In times when ML is struggling to enhance its transparency, fairness (even when the bias is in the data and not in the analysis), and accountability [35], to be auditable, to increase trust and trustworthiness among ML developers and non-ML users alike [33, 34, 36],

[1] Unfortunately, most comparisons of BNCs are among themselves [18, 23, 24, 25, 26, 27] and not to non-BNCs.

[2] Conventional categorization of interpretable ML methods [28] is through analysis of model components using, for example, linear regression and decision trees, sensitivity studies of input perturbations, or analysis of local or global surrogate approximations of the ML model [29]. Although these methods show readiness and stability, many challenges, such as dealing with dependent features, causal interpretation, and uncertainty estimation remain. These challenges need to be resolved for successful application of interpretable ML methods to scientific problems [28, 30].

[3] This is not necessarily true for all ML tools, but DNNs, following their recent meteoric rise, attract this criticism on behalf of the entire ML field, as DNNs can justifiably be considered black boxes even for ML specialists.

and even to be responsible [32],[4] researchers do their utmost to develop dedicated schemes that can help explain the predictions and decisions made by ML models [30, 37]. One example of this is the SHapley Additive exPlanations (SHAP [38]) and local interpretable model-agnostic explanations (LIME [29]) for linear models and ensemble classifiers. Another example is the tremendous effort of researchers to enhance DNN transparency, visualizability, and explainability. This may be done by advancing visualization methods as a fundamental building block that, combined with additional tools, will empower humans to understand DNNs [39], by generating saliency maps that indicate the relevance of image pixels to the network output [40, 41], or building inference graphs to interpret hidden layer activity for understanding the general inference process of a class, as well as explaining decisions the network makes regarding specific images [42]. While these schemes enrich our explainability tools, they have only limited causal interpretations [43]. A causal explanation for the mechanism of a DNN gives insight about the meaning of the DNN's output, its relation to the network input, and any change in it. In highly sensitive domains involving peoples' lives, company finances, criminal justice, and autonomous driving, a causal explanation is critical to creating justification, transparency, trust, and eventually co-operation. The ultimate goal is that any such causal explanation will be accessible and comprehensible to a human, who may then challenge the explanation until it is fully understood.

The BNC is a natural means for knowledge representation. Its graphical structure [1, 2] on the one hand and causal inference mechanisms [44, 45] on the other hand readily convey interpretability and explainability. First, the BNC provides a feature-selection mechanism through its Markov blanket (MB).[5] Conditioned on its MB, a variable is independent of the rest of the nodes in the network, allowing us to focus our attention on exploring the importance of interrelations of this variable only with those in its MB. This gives a BNC an advantage over conventional feature (variable)-selection and importance ranking methods that can only analyze variables separately (or in simple interactions), and lack any ability to consider their interrelations. Moreover, the BNC demonstrates a hierarchical structure of interrelations among both MB and non-MB variables, allowing us to explore and understand the source of the feature importance (e.g., by being included in the MB) and to identify causal paths of influence[6] originating, passing, or ending in/through

[4] Interpretability, explainability, transparency, fairness, accountability, auditability, trustworthiness, and responsibility—can we wholeheartedly confirm that we have demanded all of these from ourselves as human decision-makers in the era prior to machine learning? I will leave this question open until the discussion.

[5] The MB of a network node (representing a domain variable) includes the node, its parents, children, and children's co-parents.

[6] Implicit paths that may become explicit in the absence of latent variables in the domain or following intervention [44]. Nevertheless, experimental studies exercising intervention are hard to make and follow, whereas observational approaches such as those exerted by the BN are easy to make, follow, and interpret either by assuming no latent mechanisms in the domain or by learning these mechanisms from data (see Sect. 4).

a variable [1, 2, 44]. Regarding classification, the MB of the target (class) variable fosters discriminability in a lower dimension by enabling a quick exploration of those causal relations within the blanket that contribute to accurate classification [46, 47, 48, 49, 50, 51, 52]. Simultaneously, MBs of selected non-target variables allow deeper exploration and understanding of mechanisms establishing the domain and thereby also promote interpretability and explainability. Like using the MB, a BNC may also allow investigation of why an instance (pattern) has been classified positively or negatively by identification of a minimal set of the currently active features responsible for the current classification, or a minimal set of features whose current state (active or not) is sufficient for the classification [53]. Moreover, the BN enables inference on the values of any combination of variables (related or not to the classification) conditioned on values of the remaining variables (or only those in a specific MB) [46]. Finally, studying intervention as a source of causality is natural only to using the inference tools of graphical models [44, 45]. That is, BNs, BNCs, and graphical models, in general, have tremendous potential to promote explainable AI, and the ML community is encouraged to divert some of its efforts in this direction. We will make the argument for this later in the chapter, specifically in Sects. 3.4 and 4.

However, in order to also provide interpretability and explainability in domains having non-semantic huge (in space and/or time) inputs such as those found in images, speech, and text, much progress in BN learning algorithms is needed. Today, neither BNs nor BNCs can cope with high-dimensional raw data coming from images, speech, and text, and thus the contribution of these networks, specifically to advance interpretability of DNNs, can only come when applied to already processed, projected, or embedded raw data. We will address this in the conclusion.

In Sect. 2, we describe the most popular, though restricted, BNC, the naïve Bayesian classifier, and some of its many variants. In Sect. 3, we survey general unrestricted BNCs, focusing on the risk minimization by cross-validation BNC, and in Sect. 4, we extend our study to causal–temporal BNs and their application to classification. Section 5 concludes the chapter.

2 Restricted BNCs–The Naïve Bayesian Classifier and Its Variants

Although they make assumptions about the domain that are sometimes found unjustified (variables are class-conditionally independent [54], a maximal number of node parents [2], existence of a variable topological order [4]), restricted BNCs are surprisingly accurate as well as efficient, as structure learning is avoided or dramatically shortened. In this chapter, we will focus on and review the naïve Bayesian classifier (NBC), which is the most fundamental restricted BNC, continue by demonstrating variants of the NBC, which try to relieve the class-conditional independence assumption of the NBC, and conclude by presenting a new variant of the NBC demonstrating empirical advantage.

2.1 Naive Bayesian Classifier

Although outdated, the NBC [54]—a learning-free structure, which is obtained with virtually no computational effort—is considered a simple, practical, and state-of-the-art classifier [9, 13, 18], very often selected as the default for pattern classification in industrial applications.

The NBC is a special case of a BN consisting of a finite set of random variables, $U = \{X_1, X_2, \ldots, X_m, C\} = \{\boldsymbol{X}, C\}$, where $X_1, \ldots, X_m$ are the observable variables that represent the problem features and C is the class variable having K states. The NBC is termed naïve since it makes use of a simplifying assumption that its observable variables are conditionally independent given the class variable. All edges of the NBC are directed from the class variable to the observable variables (Fig. 1); hence, the only parent of the observable variables, X_i, is $Pa_i = C$, and $Pa(C) = \emptyset$ for the class variable.

Given a selected network structure, the NBC assigns a test pattern $\mathbf{x}$ to the class C_k ($k = 1, \ldots, K$) with the highest a posteriori probability

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)P(C_k)}{p(\mathbf{x})}, \tag{1}$$

where $p(\mathbf{x}|C_k)$ is the class-conditional probability density, $P(C_k)$ is the a priori probability for class C_k, and $p(\mathbf{x})$ is the unconditional density normalizing the product of the two probabilities such that $\sum_k P(C_k|\mathbf{x}) = 1$. The NBC independence assumption eliminates the "curse of dimensionality" since density estimation requires only linearly rather than exponentially increasing numbers of patterns. Omitting $p(\mathbf{x})$, which is common to all classes, the posterior probability can be written as

$$P(C_k|\mathbf{x}) \propto p(\mathbf{X} = \mathbf{x}|C_k)P(C_k) = P(C_k)\prod_{i=1}^{m} p(X_i = x_i|C_k), \tag{2}$$

where $\mathbf{X} = \mathbf{x}$ represents the event that $X_1 = x_1 \wedge X_2 = x_2 \wedge \ldots \wedge X_m = x_m$ and $\prod_{i=1}^{m} p(X_i = x_i|C_k)$ is a product of class-conditional densities for $\mathbf{x}$.

Both $P(C_k)$ and $p(X_i|C_k)$ can be estimated from the training data. $P(C_k)$ is the relative frequency of patterns belonging to C_k out of all of the patterns

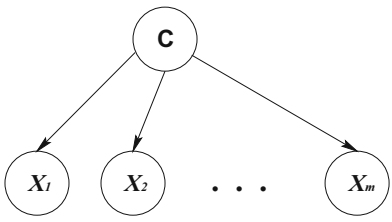


Fig. 1 The NBC depicted as a BNC in which the observable variables ($X_1, X_2, \ldots X_m$) are conditionally independent given the class variable (C)

in the data (the class prior probabilities). To estimate $p(X_i|C_k)$ (or $p(x_i|C_k)$), the one-dimensional class-conditional probabilities (or densities) for discrete (or continuous) variables, for each class C_k and variable X_i, we employ a training set comprising of a finite number of patterns $\mathbf{x}^n$, where n gets values for each of the N_k training patterns of class C_k. For a discrete variable, the class-conditional probability is estimated using the sample (unsmoothed or smoothed) frequency of each value of the variable [6, 9, 25]. For a continuous variable, the parameters of $p(x_i|C_k)$ are usually estimated by maximum likelihood using any of the popular density estimation methods such as single Gaussian estimation, which assumes the data are generated from a single normal distribution, kernel density estimation models, representing the data using a linear combination of kernels around each of the training patterns, or the Gaussian mixture model, which estimates the data using a few Gaussians with adaptable parameters [55, 56].

The NBC is an interpretable model because of the (conditional) independence assumption; it is very clear how much each feature contributes toward a certain class prediction, since we can interpret the conditional probability. When the degree of independence between variables is high and the naïve assumption is justified and/or the database is small, appropriate for the small-scale learning problem of only classifier parameters, an NBC provides an accurate classifier as was demonstrated, for example, in diagnosing genetic abnormalities [56, 57, 58].

2.2 Variants of the NBC

The NBC is based on the assumption that all attributes (variables) are mutually independent, conditioned on the class attribute. On the one hand, this assumption ignores attribute dependencies and is thus often violated. On the other hand, learning from data, a BNC that can represent arbitrary attribute dependencies is intractable (Sect. 3). Thus, researchers have focused their attention on improving the NBC, which has led to many effective and efficient leaning algorithms [9, 26, 49, 59, 60, 61, 62, 63]. These may be divided into those of *feature selection* (learning an NBC based on a subset of the features that better satisfy the independence assumption), e.g., for cycle-time key factor identification and prediction in semiconductor manufacturing [64], *local learning* (learning an NBC based on a local training set rather than the whole set), *structure extension* (learning an NBC also representing dependencies among some features), *data expansion* (learning an NBC using a training set expanded from the original), and *multinet classifiers* (learning a classifier to each class separately).

For example, lazy learning algorithms are popular local learning methods for extending the NBC. Lazy learning delays learning until classification time by storing training data and waiting until it is given a test instance; that is, generalization is delayed until test time, generating a hypothesis for each instance instead of generating one hypothesis for all instances. Among lazy learning algorithms, we find the lazy Bayesian rule (LBR) [60] and selective neighborhood naïve Bayes

(SNNB) [61]. Another algorithm is the locally weighted naïve Bayes (LWNB) [62]. In LWNB, the k-nearest neighbors of a test instance are first found, and each of them is weighted in terms of its distance from the test instance. Then a local NBC is trained using the locally weighted training instances. Although it is a k-related algorithm, its classification performance is not particularly sensitive to the size of k as long as it is not too small.

Among numerous proposals to improve the accuracy of NBC by structure extension, the one-dependence estimator (ODE) is similar to the NBC except each attribute is allowed to depend on at most one other attribute, in addition to the class attribute. The ODE provides a simple, yet powerful alternative to NBC. Its most popular variant is the tree-augmented network (TAN) [9] that uses the Chow-Liu algorithm [65] to learn a maximum weighted spanning tree over all non-class variables that are connected pairwise by edges weighted by the conditional (on the class variable) mutual information between these variables. The super-parent (SP) TAN algorithm (SP-TAN) [26] greedily learns a TAN from NBC by adding in each iteration the edge achieving the highest (cross-validation) accuracy improvement. It demonstrates remarkable classification performance but at a considerable computational cost. The averaged one-dependence estimator (AODE) [59] weakens the NBC attribute independence assumption by averaging all SuperParent-one-dependence estimators (SPODEs) [26] that satisfy a minimum support constraint, where a SPODE allows each attribute to depend on a common single attribute (i.e., SP) in addition to the class. This technique achieves comparable classification accuracy to SP-TAN with a substantially improved computational efficiency at training time. In ensemble selection of SPODEs, we select only some of the SPODEs that are averaged by AODE. This improves the classification accuracy while reducing the classification runtime, albeit at a cost of additional training time. K-dependence Bayesian (KDB) and selective KDB (SKDB) [22] classifiers allow every variable to be conditioned on the class and, at most, k other attributes. SKDB classifiers showed an advantage over the NBC and TAN. Other methods may initialize a structure search procedure, such as the K2 algorithm [4], using the NBC in order to extend the naïve classifier using more meaningful connections among graph nodes that may improve its performance [66].

The Bayesian multinet classifier (BMC) is another powerful extension of the NBC [9, 67]. A BMC comprises a set of local networks, each corresponding to a value that the class node can take. While a BNC forces the relations among the attributes to be the same for all values of the class node, a BMC allows these relations to be different for different values of the class node, forming a local network for each class and thereby providing a more expressive representation than, for example, the NBC and TAN. Conventionally, each local network is learned by minimizing the KL divergence (also maximizing the log likelihood [9]) to induce a Chow–Liu (CL) tree [65]. Using the estimated class prior probabilities, the BMC classifies to the class maximizing the product of the prior and the variable joint probability for this class estimated using only the class patterns. Although a local network must be searched for each class, the BMC is generally more accurate and has a smaller computational complexity than a BNC because each local network

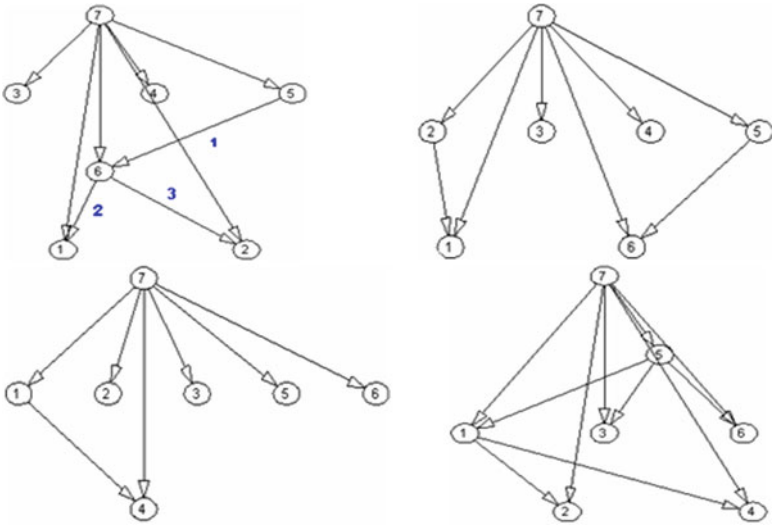


Fig. 2 Four local networks learned by the $t\text{BCM}^2$ algorithm for the four classes of the UCI Car database: C_1 (top-left), for which the order of edge learning is indicated, C_2 (top-right), C_3 (bottom-left), and C_4 (bottom-right). Node 7 is the class variable [69]

has a simpler problem to model, using a lower number of nodes in both a static scenario [67] and a dynamic one [68]. The BMC has two flaws [69]. The first is that constructing a CL tree using joint-probability-based scores for evaluating a structure is less specific to classification, i.e., CL multinet classifiers based on structures providing high scores are not necessarily accurate. The second flaw is that training a local network is based only on patterns of the corresponding class. Although this approach may approximate the class data effectively, information discriminating between the class and other classes may be discarded, undermining selection of the structure that is most appropriate for distinguishing this class. The TAN-based Bayesian class-matched multinet ($t\text{BCM}^2$) [69] utilizes a discrimination score for each local network separately, which maximizes accuracy by simultaneously detecting and rejecting patterns of the corresponding class and other classes, respectively, using both the entire data set, and the SuperParent algorithm to learn the TAN that maximizes this discrimination score. The $t\text{BCM}^2$ demonstrated [69] superiority over the naïve Bayesian, TAN, CL multinet, and recursive Bayesian multinet (RBMN) [70] classifiers for 32 UCI [71] databases. Figure 2 shows an example of a BMC learned using $t\text{BCM}^2$ for the UCI Car database.

2.3 Experimental Evaluation of NBC Variants

While [72] select the same ensemble of SPODEs (SuperParent-one-dependence estimators [26]) for all the classes, the multi-class SPODE (MSPODE) we present

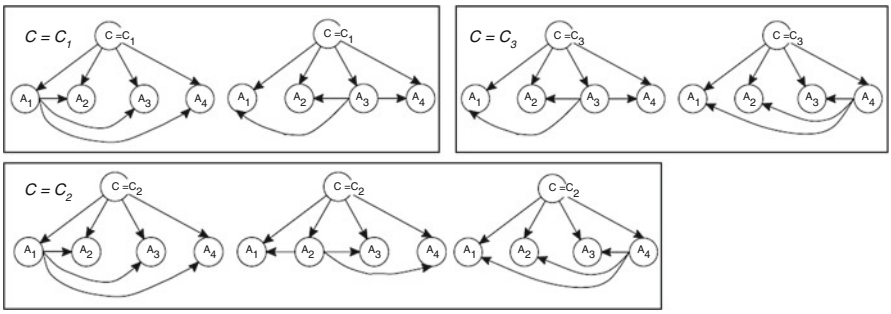


Fig. 3 An example MSPODE for a four-dimensional three-class classification problem. Note the existence of a super parent in each local network of an SPODE

here, inspired by the BMC, selects a different ensemble of SPODEs for each class (Fig. 3). An MSPODE may be learned for each class using the conditional mutual information [1] or detection–rejection measure [69]. This classifier has been analyzed theoretically and is empirically evaluated here in Table 1 in comparison to other BNCs using 32 UCI [71] data sets in terms of classification accuracy, and training and test times. Table 1 shows that although all conventional approaches are effective, accuracy of the MSPODE is superior with a similar time complexity.

3 Beyond the NBC—the Unrestricted BNC

NBC and its variants often lack not only accuracy, but also interpretability and explainability due to the naïve assumptions they make and the limited structure spaces they search. However, along with the traditional classifiers based on the neural network (NN), support vector machine (SVM), decision tree, and ensembles (e.g., the RF and XGBoost family), classifiers based on the BN (and not the NBC) have been introduced and studied for 25 years [9, 10, 11, 12, 13, 14, 15, 16, 17, 25, 27, 47, 48, 66]. To promote the use of the BNC, we first present the general BN (Sect. 3.1) on which the BNC is based. We then outline the conventional (likelihood-driven) unrestricted BNC (Sect. 3.2), before introducing the risk minimization by cross-validation (RMCV) BNC (Sect. 3.3).

3.1 The General BN

A BN model $\mathcal{B}$ for a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$, each having a finite set of mutually exclusive states, consists of two main components, $\mathcal{B} = (\mathcal{G}, \Theta)$. The structure $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a directed acyclic graph (DAG). $\mathbf{V}$ is a finite set of nodes of $\mathcal{G}$ corresponding to $\mathbf{X}$ (usually, X_i refers to both the variable

Table 1 Accuracies (%) and time complexities of our suggested multi-class SPODE (MSPODE) vs. popular variants of the NBC classifying 32 UCI databases. Besides the NBC, the variants are: Averaged one-dependence estimator (AODE) [59], lazy AODE (LAODE) [73], one ensemble selection of SuperParent-one-dependence estimators (ESPODE) [72], and tree-augmented network (TAN) [9]. k, t, n, and v are the numbers of classes, training instances, features, and feature values (average), and s is the average similarity value between a test instance and each training instance [73]

Database	NBC	AODE	LAODE	ESPODE	TAN	MSPODE
Australian	85.2	85.6	85.0	**86.4**	84.0	**86.4**
Balance	**90.9**	87.2	88.1	85.2	87.3	88.7
Bupa	64.7	66.5	67.1	66.5	**67.7**	66.5
Car	86.1	88.4	88.5	88.7	**94.3**	90.4
Corral	86.7	93.3	95.0	**98.3**	97.9	**98.3**
Crx	86.1	**89.2**	**89.2**	**89.2**	85.9	**89.2**
Cytogenetics	78.1	82.3	82.1	80.2	81.3	**83.4**
E.coli	86.4	83.3	85.9	84.9	**86.9**	85.4
Flare	82.0	**86.5**	86.2	**86.5**	85.6	**86.5**
Hayes	79.5	82.5	**83.1**	78.1	76.7	79.1
Hepatitis	70.0	72.5	72.5	73.7	72.5	**73.8**
Ionosphere	91.7	91.7	**93.1**	91.1	92.3	92.0
Iris	94.7	94.7	94.0	93.3	94.3	**96.7**
Krkp	88.4	91.9	N/A	93.5	**94.3**	93.5
Led-7	74.6	74.8	N/A	74.5	74.0	**75.5**
Lymphography	85.2	86.2	82.1	86.2	80.7	**88.0**
Monks	96.4	**98.9**	98.7	98.4	97.3	98.7
Nursery	90.2	92.7	N/A	92.4	93.4	**93.7**
Pendigit	87.3	97.6	N/A	**97.7**	95.7	97.6
Pima	76.0	75.9	75.1	75.4	75.5	**76.5**
Post-operative	67.5	68.7	70.0	70.0	**71.2**	70.0
Segment	92.1	95.4	**97.0**	95.7	94.4	96.4
Shuttle	98.7	99.8	N/A	99.8	**100**	99.8
Splice	94.8	95.5	N/A	95.5	88.8	**96.8**
Tic Tac Toe	69.4	75.8	**81.8**	74.5	74.7	74.5
Tokyo	91.3	**94.2**	93.9	93.3	91.9	93.5
Vehicle	61.5	72.5	72.9	72.7	70.0	**73.0**
Voting	91.3	**96.1**	**96.1**	94.8	94.4	95.7
Vowel	66.7	87.2	90.8	89.3	83.5	**91.1**
Waveform-21	81.7	84.5	N/A	82.8	82.0	**85.4**
Wine	**98.8**	97.7	97.1	97.1	98.2	**98.8**
Zoo	93.0	93.0	**96.0**	93.0	**96.0**	94.0
Average accuracy	84.0	86.9	86.4	86.8	86.3	**87.7**
Training complexity	$O(tn)$	$O(tn^2)$	$O(tn^2)$	$O(ktn^2)$	$O(n^2(t+kv^2))$	$O(ktn^2)$
Test complexity	$O(kn)$	$O(kn^2)$	$O(stn^2)$	$O(kn^2)$	$O(kn)$	$O(kn^2)$

[a] Bold font is for most accurate classifier for a task

and its corresponding node), and $\mathbf{E}$ is a finite set of directed edges of $\mathcal{G}$ connecting $\mathbf{V}$. Edges and missing edges encode dependencies and conditional independencies, respectively, in $\mathcal{G}$. Θ is a set of parameters that quantify the structure. The parameters are local conditional probability distributions (or densities), $P(X_i = x_i|\mathbf{Pa}_i, \mathcal{G})$, for each $X_i \in \mathbf{X}$ conditioned on its parents in the graph, $\mathbf{Pa}_i \subset \mathbf{X}$. Most studies, including this one, focus mainly on discrete variable BNs and complete data.

The joint probability distribution over $\mathbf{X}$ given $\mathcal{G}$—assumed to encode this distribution—is the product of these local probability distributions [1, 2, 4, 5],

$$P(\mathbf{X} = \mathbf{x}|\mathcal{G}) = \prod_{i=1}^{n} P(X_i = x_i|\mathbf{Pa}_i, \mathcal{G}), \tag{3}$$

where $\mathbf{x}$ is the assignment of states to the variables in $\mathbf{X}$ and x_i is X_i's state.

During inference, the conditional probability distribution of a subset of nodes in the graph (the "hidden" nodes) given another subset of nodes (the "observed" nodes) and the BN model is calculated. A common method for exact inference is the junction tree algorithm [3], but when there is only one hidden node (e.g., the class node in classification), direct inference based on Eq. 3 and Bayes' rule is more feasible. Note that the computation of conditional probability distributions for inference depends on the graph. Thus a structure, either based on expert knowledge or learned from the data, must first be obtained.

The search-and-score (S&S) approach to learning a structure from data [4, 5, 6] comprises a search for the structure achieving the highest score, e.g., hill climbing (HC), and a score, generally the Bayesian score,

$$P(\mathcal{G}|D) = \frac{P(D|\mathcal{G})P(\mathcal{G})}{P(D)} = \frac{P(D, \mathcal{G})}{P(D)} \tag{4}$$

for a structure $\mathcal{G}$ given a data set $D = \{v_1, v_2, \ldots, v_N\}$, which is a random sample of N independent patterns from the joint probability distribution of $\mathbf{X}$.

3.2 The General BNC

While the BN provides a powerful graphical model for encoding the probabilistic relationships among a set of variables and can therefore naturally be used for classification, BNCs learned in the common way using likelihood scores usually tend to achieve only mediocre classification accuracy because these scores are less specific to classification, but rather suit a general inference problem. Learning a BNC requires learning the structure (graph) of the graphical model and its parameters so that the learned BN will excel in inference of a specific variable—the class variable—and not necessarily of all variables. When focusing on structure

learning, exhaustively searching the space of possible graphs is infeasible [4], and thus S&S structure learning algorithms sub-optimally search the space and select the structure achieving the highest value of a score [4, 5, 6]. However, until very recently, all S&S structure learning algorithms used a generative score and thereby led to learning a generative model that is not specific to classification, but to general inference. Researchers have demonstrated that BNC structures learned using generative scores do not usually contribute to high classification accuracy [9, 10, 11, 12, 16, 17] since there is lack of agreement between these scores used for learning and the score used for evaluation, which is the classification accuracy. That is, classifiers based on structures having high generative scores are not necessarily highly accurate.[7]

It is clear from Eq. 4 that a score should reflect a correspondence between the structure and the data. The minimum description length (MDL) score [74] can approximate $P(D|\mathcal{G})$—the marginal likelihood [5, 6]—but [9] argued that this score is not suitable for classification and instead recommended the class-conditional log likelihood (CLL) (as opposed to log likelihood (LL)),

$$CLL(\mathcal{G}|D) = \sum_{i=1}^{N} \log P(c_i|v_i'), \tag{5}$$

where the vector v_i for the ith instance in D consists of a feature vector v_i' and a class label c_i, so that $v_i = (c_i, v_i')$. Notice that $CLL(\mathcal{G}|D) = \sum_{i=1}^{N} \log P(v_i) - \sum_{i=1}^{N} \log P(v_i') = LL(\mathcal{G}|D) - \sum_{i=1}^{N} \log P(v_i')$.

By maximizing CLL, the structure that best approximates the probability of predicting the class given feature values for every pattern is learned [10]:

$$P(c^N|v'^N, \mathcal{G}) = \frac{P(c^N, v'^N|\mathcal{G})}{P(v'^N|\mathcal{G})} = \frac{P(D|\mathcal{G})}{\sum_{c'^N} P(c'^N, v'^N|\mathcal{G})}, \tag{6}$$

where v'^N consists of all feature vectors and c^N consists of all possible combinations of the r_C states of the class variable C in a random sample D of size N. The computation of this score is infeasible, since the sum in the denominator is exponential in N (r_C^N terms), let alone score maximization.

An approximation [10] considers the left-hand side of Eq. 6 and the marginal likelihood Eq. 4 as the supervised and unsupervised marginal likelihoods, respectively. The marginalization over the parameters in Eq. 6 is

$$P(c^N|v'^N, \mathcal{G}) = \int_{\Theta} P(c^N|v'^N, \Theta, \mathcal{G}) P(\Theta|v'^N, \mathcal{G}) d\Theta, \tag{7}$$

[7] Note, however, that although constraint-based structure learning algorithms of BNs [2] are usually not considered in inducing a BNC, they may nevertheless lead to supreme BNCs [27].

and its approximation [10] using a single term is

$$P(c^N|v'^N, \mathcal{G}) \approx P(c^N|v'^N, \hat{\Theta}, \mathcal{G}). \tag{8}$$

$\hat{\Theta}$ is the parameter configuration maximizing the parameter posterior probability, $P(\Theta|v'^N, c^N, \mathcal{G})$, which is a different solution than that derived when maximizing $P(c^N|v'^N, \Theta, \mathcal{G})$. However, there is no general closed-form solution to the supervised form of the score, and the posterior is not decomposable in this case, hence the need for approximation [9].

Another predictive local criterion (LC) [5] for learning a BNC [10] is based on the prequential approach [75],

$$LC(D, \mathcal{G}) = \sum_{i=1}^{N} \log P(c_i|\{v_j\}_{j=1}^{i-1}, v'_i, \mathcal{G}). \tag{9}$$

Other cumulative logarithmic loss scores [10] use 10-fold cross-validation (CV) or leave-one-out, which are reputable methods for model selection [76]. They are both described here under the general term CV-K, where $K = 10$ or $K = N$ for the two cases, respectively. A score using CV-K for predicting a class is defined:

$$CV_K(D, \mathcal{G}) = \sum_{k=1}^{K} \sum_{i=1}^{N/K} \log P(c_{i+A_k}|D \setminus D_k^K, v'_{i+A_k}, \mathcal{G}), \tag{10}$$

where $A_k = (k-1)N/K$ and $D_k^K = \{v_{j+A_k}\}_{j=1}^{N/K}$ is a validation set derived from the training set D.

Using either of the supervised (conditional) marginal likelihood scores (Eqs. 7, 9, or Eq. 10) for learning a BNC is asymptotically optimal. However, for a finite sample, though a high score value may indicate correct classification, it cannot guarantee it.

A score that measures the degree of compatibility between a possible state of the class variable and the correct class is the 0/1 loss function:

$$L(c_i, \widehat{c_i}) = \begin{cases} 0, & c_i = \widehat{c_i} \\ 1, & c_i \neq \widehat{c_i} \end{cases}, \tag{11}$$

where c_i is the true class label and $\widehat{c_i}$ is the estimated class label for the ith instance.

To demonstrate the difference between a class-conditional score and the 0/1 score, consider a two-class classification problem, two candidate classifiers A and B, and two instances v'_1 and v'_2 [17]. Classifier A predicts the correct class for instances v'_1 and v'_2 with probabilities of 0.3 and 0.51, respectively, while classifier B predicts the correct class for the same two instances with probabilities of 0.45 and 0.49. Since the sum of log probabilities (i.e., "log-loss" score) is larger for classifier B than for classifier A, the former classifier will be selected. However, if evaluating

the 0/1 loss values, classifier B is inaccurate for both v'_1 and v'_2, whereas classifier A is correct for v'_2. Thus, choosing classifier A based on the 0/1 loss score is more sensible for classification than choosing classifier B based on the log-loss score. We therefore suggest using the classification-specific 0/1 loss function for learning BNCs of enhanced classification accuracies.

3.3 Risk Minimization by Cross-Validation

We propose risk minimization by cross-validation (RMCV) for a classification-oriented score and an S&S algorithm for learning unrestricted BNCs. Note that other uses of classification-oriented scores in learning unrestricted BNCs [12, 13] are in a somewhat different context. While commonly used S&S algorithms use likelihood-based scores suitable for general inference, RMCV minimizes an empirical estimation of the classification error rate and thereby learns highly accurate BNCs. This model does not need to estimate the true distribution, generate data from this distribution, or infer about any non-class variable. That is, RMCV performs discriminative learning of a generative (BN) model. It learns generative models that are complicated, only to discriminate accurately among classes. The RMCV uses the 0/1 loss function, which is a classification-oriented score for unrestricted BNCs and non-BN classifiers alike. Its superiority to marginal and class-conditional likelihood-based scores with respect to classification accuracy using small real and synthetic problems, allowing for learning all possible graphs, was empirically demonstrated [17].

Instead of selecting a structure based on summation of supervised marginal likelihoods over the data set (Eq. 9 or Eq. 10), we suggest selecting a structure based on summation of false decisions about the class state over the data set. Our score is based on risk minimization [77] using the 0/1 loss function measured on a validation set. The training set D is divided into a validation set D^K (having N/K of N instances) and an effective training set (having $N(K-1)/K$ instances). The classification error rate (0/1 loss) is measured for each candidate structure and in any iteration of the search on D^K. During learning, no use of a (third) test set is made. As part of a CV experiment, the score of a candidate structure is computed by averaging the error rates over K non-overlapping validation sets. Since the structure that minimizes the empirical risk is being searched for, the score is called risk minimization by cross-validation (and we deliberately do not simplify $\frac{1}{K}\frac{K}{N}$) [17]:

$$RMCV_K(D,\mathcal{G}) = \frac{1}{K}\sum_{k=1}^{K}\frac{K}{N}\sum_{i=1}^{N/K} L(c_{ki}, \arg\max_{c\in\{c_1,\ldots,c_{r_C}\}} P(C=c|D\setminus D_k^K, v'_{ki}, \mathcal{G})), \tag{12}$$

where $v_{ki} = (c_{ki}, v'_{ki})$ is the ith instance of D_k^K and $L(\,,\,)$ is the 0/1 loss function (Eq. 11). Being a CV-based score, RMCV is easy to implement and computationally

feasible (see, for example, Eq. 8), and it depends on only one parameter (K). Further, it is argued [10] that a CV-based score can be regarded as an approximation of a factorization of the supervised marginal likelihood (Eq. 6). Note that the RMCV score is normalized by the data set size N, whereas Eqs. 9 and 10 are not. Although normalization has the same effect on all learned structures, it can clarify the meaning of the score (i.e., an error rate) and help when comparing scores over data sets. Moreover, sharing the same range of values ([0, 1]), RMCV establishes its correspondence to classification accuracy.

To compute RMCV, the candidate structure has to be turned into a classifier by learning its parameters. Local probabilities are modeled using the unrestricted multinomial distribution [5] (assuming discrete variables), where the distribution parameters are obtained using maximum likelihood [4], similar to [10, 25]. Moreover, [11] argued, based on experiments, that maximum likelihood parameter estimation does not deteriorate the results compared to maximum conditional likelihood estimation, which can only be obtained by computationally expensive numerical approximation. Learning a BN rather than a structure has an additional cost of parameter learning, though this cost is negligible when using maximum likelihood estimation and fully observed data.

To prevent over-fitting the training set, RMCV is computed by K-fold CV. Thus, over-fitting is controlled through the score itself, and not through the search dynamics as in other algorithms discussed here. Also, note that the same measure is used for learning the BNC and for evaluating its accuracy, which makes learning oriented toward classification. Similar to CLL-based scores [10, 11], RMCV is not decomposable.

A suggested S&S structure learning algorithm consists of the RMCV score and a simple HC search [17]:

Algorithm RMCV **Input:** An initial DAG, $\mathcal{G}$; A training set that is partitioned to K mutually exclusive validation sets, $D = \{D_k^K\}_{k=1}^K$. **Output:** BN model ($\mathcal{G}$,Θ).

compute $RMCV_K(D, \mathcal{G})$
converged:= false
While converged = false
  For each $\mathcal{G}' \in$ Neighborhood($\mathcal{G}$)
    compute $RMCV_K(D, \mathcal{G}')$
  $\mathcal{G}^* := \arg\min_{\mathcal{G}'} RMCV_K(D, \mathcal{G}')$
  If $RMCV_K(D, \mathcal{G}^*) < RMCV_K(D, \mathcal{G})$
    Then $\mathcal{G} := \mathcal{G}^*$
    Else converged:= true
Θ:=LearnParameters($D, \mathcal{G}$)
Return ($\mathcal{G}$, Θ)

The RMCV algorithm starts with any initial graph (e.g., the empty graph or the NBC) and a training set that is divided into K mutually exclusive validation sets. For each $k \in \overline{1, K}$, the parameters are learned using the effective training set $D \setminus D_k^K$, and the error rate is evaluated using D_k^K. The average error rate over the K validation

sets $D_k^K, \forall k \in \overline{1, K}$ is the RMCV score (Eq. 12). The initial graph and its score are kept as the current graph and score, respectively. Next, the neighborhood of the current graph is generated by all single edge additions, deletions, and reversals. Since only DAGs are allowed, any cyclic directed graphs in the neighborhood are excluded [78]. The graph having the lowest RMCV score in the neighborhood is chosen. Its score is compared to the current score, and the search is halted if the current score is lower than or equal to the score of the chosen graph. If, however, the chosen graph has a lower score than the current score, it becomes the current graph and the procedure repeats itself. During structure evaluation, only the effective training sets $D \setminus D_k^K, \forall k \in \overline{1, K}$ are used for parameter learning. Yet, once the search for a structure completes, the role of the validation sets is finished and the entire training set D can be used for more reliable parameter learning for this structure, rendering the structure a BN. The algorithm then returns the learned BN defined by $(\mathcal{G}, \Theta)$.

Note that when using maximum likelihood parameter estimation, fully observed data, and the suggested search, there is no need to reassess all of the parameters for the different structures during each HC step. Parameters are changed only for nodes whose set of parents has been modified. In case of an addition or deletion of an edge, only one node is affected, and in case of a reversal of an edge, only two nodes are affected. For the same reason, it is beneficial to keep the history of the probability calculations, using the factorization of Eq. 3, for the initial/current DAG.

3.4 Experimental Evaluation of Unrestricted BNCs

Our first evaluation (Sect. 3.4.1) follows previous research that conventionally uses synthetic and traditional data sets. The empirical investigation includes several unrestricted BNCs. The RMCV algorithm is generally initialized by either the NBC or an empty structure to induce the RMCV (NBC) or RMCV (Empty) BNCs, respectively [17]. Along with the NBC, these RMCV BNCs are compared here with three other types: BNCs learned using the K2 score and algorithm [4] initialized by either the NBC or empty structures [66], i.e., K2 (NBC) and K2 (Empty); BNCs learned using HC search initialized by either the NBC or empty structures to minimize the MDL score [74] (maximize the marginal likelihood), i.e., MDL (NBC) and MDL (Empty) [17]; and BNCs learned to maximize the class-conditional log likelihood, initialized by the empty graph, which use either HC search or a two-parent limitation on a node, i.e., BNC-MDL and BNC-2P [11].

Our second evaluation (Sect. 3.4.2) presents original studies using the RMCV algorithm and own authentic data sets from five real-life domains in genetic abnormality inspection, semiconductor manufacturing, Parkinson's disease diagnosis, amyotrophic lateral sclerosis (ALS) prediction and explanation, and young driver motorcycle accidents analysis.

3.4.1 Evaluation of BNCs Using Traditional Data Sets

BNCs were found comparable and even superior to non-BN classifiers in different reports. Grossman and Domingos [11] showed that when a BNC's structure is learned to optimize the conditional likelihood of the class variable (Eq. 5), it is advantageous to the classification and regression tree (CART) classifier [79]. Pernkopf [13] compared variants of the NBC and BNCs to the selective k-nearest neighbor classifier (skNN), selecting by sequential feature selection a subset of features that maximizes the classification performance. He showed using several UCI [71] databases that the BNCs are usually more accurate than the skNN and are superior with respect to memory requirements and computational demands during classification. The selective k-dependence Bayesian (SKDB) classifier [22] showed an advantage over the NBC, TAN [9], and AODE [59] (see Sect. 2.2 for details about the two later classifiers), and comparable accuracy to the RF [80], with no significant difference between them based on the Friedman test followed by the Nemenyi post-hoc test [81].

Kelner and Lerner [17] reported classification performance using 22 UCI [71] databases with various characteristics, e.g., the numbers of classes, features, and patterns between 2 and 26, 4 and 36, and 80 and 20,000, respectively. Table 2, extracted from [17], shows dominance of the RMCV over all other BNCs, and also over non-BNCs, such as the CART, three-layer-perceptron NN [82], and SVM [83] with its three conventional kernels—all non-BNCs were optimized for each task separately. According to a Friedman test followed by a Bonferroni–Dunn post-hoc test with RMCV as the control classifier vs. all other BNCs, RMCV was superior to all of them with a significance level of $p < 0.05$, with the exception of BNC-MDL, for which $p < 0.1$. According to a Friedman test for all of the classifiers, RMCV was ranked the highest, and the null hypothesis that all algorithms are the same had been rejected with high confidence. However, due to the large number of models compared (fourteen), a relatively large difference of average ranks due to the Friedman test was required by the Bonferroni–Dunn test to indicate a significant difference, and RMCV was found to be significantly superior (with $p < 0.1$) to

Table 2 Average and std of classifiers' accuracy over 22 UCI [71] databases. Bayesian network initial structures and SVM kernel types appear in brackets. Bold/italic fonts are used, respectively, for the best/worst classifiers

	RMCV (NBC)	MDL (NBC)	MDL (Empty)	K2 (NBC)	K2 (Empty)	BNC-MDL	BNC-2P
Average	**84.8**	81.5	80.9	81.0	80.9	81.4	*76.7*
Std	(3.8)	(4.1)	(4.1)	(4.3)	(4.4)	(3.8)	(4.7)
	TAN	NBC	CART	NN	SVM (Linear)	SVM (Polynomial)	SVM (Gaussian)
Average	82.5	81.3	83.8	83.6	82.4	77.3	81.8
Std	(3.9)	(3.7)	(3.8)	(4.0)	(3.6)	(3.9)	(3.9)

only four other classifiers [MDL (empty), K2 (NBC), K2 (empty), and BNC-2P]. The less conservative Wilcoxon signed-rank test finds RMCV to be superior (with $p < 0.05$) to all of the evaluated BNCs and also to SVM (polynomial). For CART, NN, and SVM (Linear/Gaussian), RMCV was not significantly superior with $p \in [0.147, 0.610]$. This means that RMCV, CART, NN, and SVM (Linear/Gaussian) were comparable in terms of classification accuracy. That study also showed that an optimized version of RMCV is faster than all unrestricted BNCs and comparable to the neural network with respect to runtime.

These comparative studies, like most studies cited here and elsewhere, usually use ready-to-use databases of what are called "real-world" problems, taken mainly from the UCI [71] and similar repositories. However, the selection of the sample databases used in each comparative study is neither complete nor standardized nor, needless to say, is it representative of real problems[8] [84]. To demonstrate BNC performance on a wider range of complexities presented in actual real-world data sets, we report in Sect. 3.4.2 on six studies with five such data sets.

3.4.2 Evaluation of BNCs Using Authentic Data Sets

The data set in the first study contains data extracted from fluorescence in situ hybridization (FISH) microscope images used in genetic abnormality inspection [55, 56, 85]. The various instances represent red and green signals, corresponding to Down and Patau syndromes, respectively. Each of the two signals can be either "real," where the syndrome can be observed in the image, or an "artifact" due to refraction and scattering of the fluorescence signal in the microscope optics, where the syndrome is not actually present. Each of 3,144 instances of signal images is represented by twelve features of the signal that are characterized by five types: size (area), shape (eccentricity, measuring the signal's similarity to an ellipse), intensity (different features measuring total, average, and standard deviation in the red-green-blue (RGB) channels), hue (maximum, average, standard deviation, difference between the maximum and average normalized by the average), and eigenfeatures, corresponding to the red and green intensity components of the signal. The class label takes one of the following values: Real Red (RR) (551 signals), Artificial Red (AR) (1,224), Real Green (RG) (594), or Artificial Green (AG) (775), forming a four-class classification problem.

Table 3 shows that the relatively high accuracy of the NBC in classifying FISH signals is attributed to its accuracy in classifying the Real Red signals, for which the variables are mostly independent given the class value, as assumed by the NBC. Therefore, we also attribute the high accuracies of the other NBC-based BNCs (K2, MDL) to the NBC initialization. While the K2-based classifiers are

[8] Check [84] that shows that the nine most popular UCI databases have between 100,000 and 400,000 hits, but the comparative studies using them provide no explanation to why these databases have been selected.

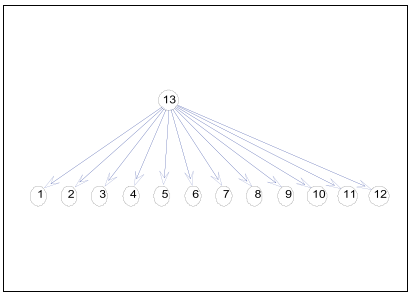
Table 3 Accuracies (%) of BNCs for four signal classification tasks for the FISH data set

Algorithm	RR	AR	RG	AG	Total
NBC [54]	*88.0*	76.9	81.1	74.4	79.0
K2 (NBC) [4]	85.8	77.6	80.1	*75.7*	79.2
K2 (Empty) [4]	75.7	72.1	81.7	71.6	74.4
MDL (NBC) [74]	85.3	80.7	**85.0**	65.8	78.7
MDL (Empty) [74]	80.2	81.1	82.0	74.3	79.4
BNC-MDL [11]	80.1	79.5	*84.0*	67.0	77.5
BNC-2P [11]	82.9	78.9	74.2	72.4	77.1
RMCV (NBC) [17]	**88.2**	*81.5*	83.7	75.1	*81.5*
RMCV (Empty) [17]	83.7	**84.2**	82.8	**76.9**	**82.1**

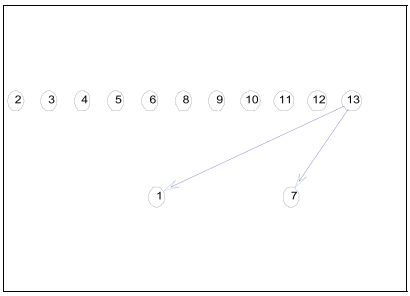
[a] Bold and italic fonts are for most and second-most accurate classifiers for a task, respectively

highly dependent on the K2 algorithm initialization, the MDL BNCs and BNC-MDL/2P are not. In all but one of the four classification tasks, at least one of the RMCV BNCs is the most or second-most accurate classifier, which makes these two BNCs the most accurate on average. Figure 4 shows (for one arbitrary fold of the CV5 experiment) that while RMCV (NBC) reveals a similar structure to that of the NBC, RMCV (Empty) shows a different structure, as six of the twelve attributes are disconnected. This aggressive but educated feature selection of RMCV (Empty) contributes not only to the best classification results (Table 3) but also to an interpretable model relying only on three types of signal features (intensity, hue, and eigenfeatures), compared to the other too dense or too sparse less informative models (Fig. 4).

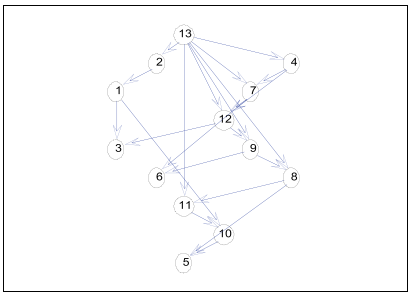
The second original data set, a flash memory semiconductor manufacturing data set, consists of 362 instances of wafer lots represented by 35 tool variables describing an ion-implementation process that is part of wafer manufacturing. The data are highly imbalanced, where 30 of the lots are faulty and 332 are normal. Table 4 shows that most BNC algorithms have been fooled by the imbalance in the data, wrongly classifying most or all of the faulty lots as normal. This is the case with K2 (but also with MDL and BNC), which are almost perfect in classifying normal lots but fail completely with faulty lots. The only two algorithms that have a relatively reasonable accuracy in classifying faulty lots in spite of the imbalance are the NBC and RMCV. The 36.7% accuracy of the NBC for faulty lots came at the expense of its accuracy in normal lots (93.1%), which is the lowest of all algorithms, positioning this classifier as the poorest in total. The RMCV, and particularly the "weighted" version, which penalizes errors according to the classes' prior probability ratio, balances its performance between the classes, positioning this classifier as the best in total and most even. Of particular note in Fig. 5 are the graphs of the NBC (best for faulty lots), K2 (NBC) (best for normal lots), and RMCV (Empty, weighted) (best in total). We attribute the high and balanced performance of the RMCV to the drastic dimensionality reduction it performed, identifying only a few significant variables in its Markov blanket—enough to accurately classify both classes but without over-fitting any of them at the expense of the other.



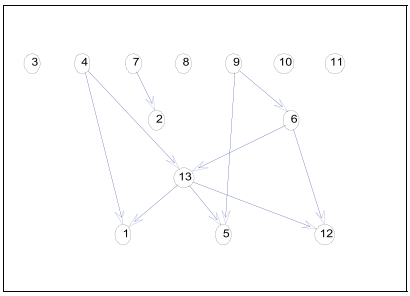
(a)



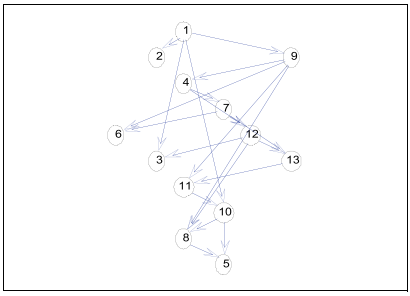
(e)



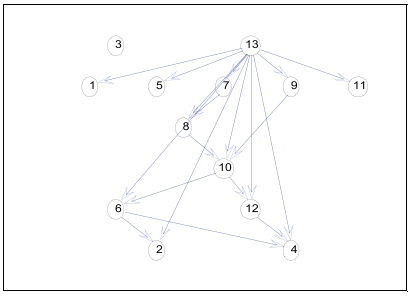
(b)



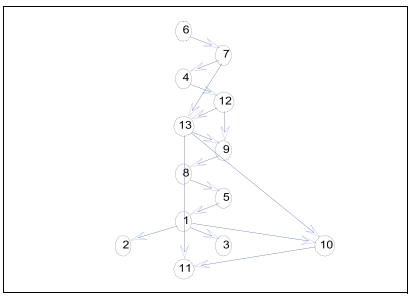
(f)



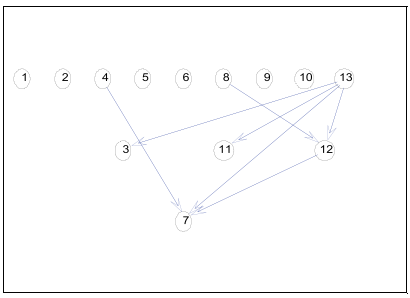
(c)



(g)



(d)



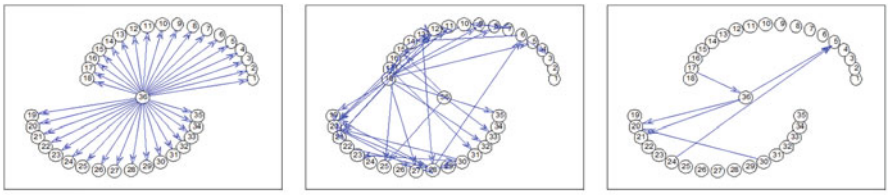
(h)

Fig. 4 BNC structures learned for one CV5 fold for the FISH data set. 13 is the class node. (**a**) NBC. (**b**) K2 (NBC). (**c**) K2 (Empty). (**d**) MDL (Empty). (**e**) BNC-MDL. (**f**) BNC-2P. (**g**) RMCV (NBC). (**h**) RMCV (Empty)

Table 4 Accuracies (%) of BNCs in detecting either normal or defective lots of wafers of an imbalanced semiconductor manufacturing data set

Algorithm	Normal	Faulty	Total
NBC [54]	93.1	**36.7**	88.4
K2 (NBC) [4]	**99.7**	0.0	**91.4**
K2 (Empty) [4]	*99.4*	0.0	*91.2*
MDL (NBC/Empty) [74]	98.2	10.0	90.9
BNC-MDL [11]	99.1	0.0	90.9
BNC-2P [11]	98.2	13.3	*91.2*
RMCV (NBC) [17]	96.7	20.0	90.3
RMCV (Empty) [17]	97.0	13.3	90.1
RMCV (NBC, weighted) [17]	97.9	13.3	90.9
RMCV (Empty, weighted) [17]	97.0	*30.0*	**91.4**

[a] Bold and italic fonts are for most and second-most accurate classifiers for a task, respectively



(a) (b) (c)

Fig. 5 Three BNCs learned for one CV5 fold for the semiconductor manufacturing data set. 36 is the class node. (**a**) NBC. (**b**) K2 (NBC). (**c**) RMCV

The data set in the third study contains medical diagnosis data extracted from visuomotor measurements of people who have been diagnosed with Parkinson's disease (PD) or essential tremor (ET) versus healthy controls. PD and ET have very similar symptoms, but ET, unlike PD, is related to long life expectancy. The data set used to predict PD has 164 instances, relatively balanced among the three classes (55 PD, 51 ET, 58 controls). Each is represented using the person's age, their worst affected hand, and fourteen visuomotor features measured for the persons' two hands. The MDL (NBC or Empty) BNC provided the highest accuracy on PD patients, and the K2 (NBC or Empty) BNC supplied the best accuracy on the healthy controls, but both classifiers failed to classify any ET patient. This failure was due to learned structures where either the class node was not connected to the graph (making classification based on the a priori probabilities) or the Markov blanket of the class node was relatively empty of nodes with which to make accurate predictions. The BNC-2P provided reasonable performances on all three classes (where the BNC-MDL was similar to the MDL and K2 classifiers). However, its scores were lower than those of the RMCV (NBC or Empty), which was equally accurate for all three classes and the most accurate classifier. The NBC, dispensing with structure learning, was the classifier least affected by the small data set; it was thus the second-most accurate classifier after the RMCV, although it manifested a

non-informative graph. The RMCV's structure was not as dense and uninformative as that of the NBC but was also not as empty as those of the MDL, K2, and BNC-MDL/2P. Such a structure conveys knowledge representation by interrelating visuomotor measurements with clinical characteristics of the patients, which also improves the accuracy in classifying PD patients, as distinct from ET patients and from healthy people.

For the fourth data set, the RMCV was applied [46] to identify the most influential variables in predicting and explaining functional deterioration (e.g., walking, writing, climbing stairs, and speaking) of five levels from a large clinical-trial database of ALS patients [86]. For each variable representing patient functionality, the RMCV selected to include in the variable's MB only those variables from the tens available for the algorithm that contribute to accurate prediction of this functionality. For example, the MB of the climbing stairs variable shows connections between the ability to climb stairs and lab test results that are related to the body muscle metabolism (e.g., glucose, creatinine, phosphorus, and alkaline phosphatase), forced vital capacity (FVC), the total amount of air a person can exhale during a forced breath, which is also related to the person's physical capability, and the disease onset site, whether in the bulbar or limbs (and then the patients are limited in climbing stairs sooner in their progression). It could also be possible to distinguish mild from severe ALS patients by the different combinations of values these MB variables take for the two groups of patients. For example, severe swallowing functionality in patients who their onset was in the bulbar and their FVC capability is low is between 8 and 35 times more frequent than in patients who their onset was in the limbs and their FVC capability is between moderate and high, respectively [46].

In the fifth study [87], when predicting young driver (YD) fatalities in motorcycle accidents, the RMCV classifier identified key factors in the class variable's MB that distinguish between minor, severe, and fatal accidents. Some of the main factors were the accident type (inexperienced YDs are more likely to lose control over their motorcycle and crash into inanimate objects, skid, or turn over with usually deadly results), road speed limit (accidents on roads where the speed limit is high tend to be fatal or severe), gender (in all fatal accidents in the data, men were the drivers, and in general, the victims of severe and fatal YD accidents are three times more likely to be men), age at time of accident (most accidents of older YDs ($\geq$22) are fatal, since they drive heavier motorcycles than younger drivers), and motorcycle type (a YD accident that involves a heavy, $\geq 400cc$, motorcycle is eight times more likely to be deadly than for a lighter one).

In the sixth and last study we report here [88], the RMCV was modified to deal with class-imbalance ordinal classification problems and to provide information about the distribution of misclassifications and about the sensitivity to error severity (distinguishing between misclassification of class X as class Y or as class Z). The modified RMCV achieved superior average accuracy over the CART, NN, and SVM using 23 synthetic and 17 UCI databases, and superior average accuracy over the RF using the synthetic databases but inferior average accuracy using the UCI

databases.[9] In addition, using data of three real problems (the above ALS [46] and YD motorcycle accidents [87], as well as missed due date—no delay, 3–5 days of delay, and more than 5 days of delay—of a product in Teleco orders), the modified RMCV classifier showed confusion matrices with errors that are the most balanced over the classes compared with the NN and SVM competitors, contributing to its superior accuracy.

Still, a further step is needed to allow examples such as those in this section—in genetic abnormality inspection, semiconductor manufacturing, Parkinson's disease diagnosis, ALS prediction and explanation, and young driver motorcycle accidents analysis—convincingly demonstrate domain experts interpretability and explainability. This step involves human–machine interaction, and we will return to this issue in the conclusion section.

4 Beyond the BNC–Causal–Temporal Classifiers

Let us consider progression of a neurodegenerative disease such as ALS or Alzheimer's or a chronic disease such as type 2 diabetes, for which we wish to predict a future state for patients. In other words, to predict a diagnosis using symptoms such as clinical markers and lab test results collected routinely over time. We may further consider a left-to-right model with a state index that either decreases or stays the same and thereby represents progression of such diseases. Figure 6a shows a two-slice graph that represents a medical domain in which there is a cause to a disease. We wish to predict the current disease state based on that cause and the previous disease state, where both the cause and disease state are unknown latent

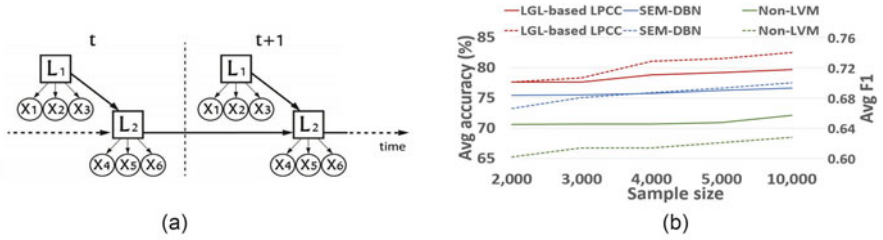


Fig. 6 (**a**) Artificial temporal graph with three OVs per each of two LVs also making a collider per slice. (**b**) Classification accuracy (solid line) and F1-measure (dashed line) for increasing sample sizes of the data sets sampled from the graph in (**a**)

[9] The modified RMCV score balances the 0/1 loss (accuracy) function with the mutual information between predictions and true labels and with the severity of misclassifications [88, 89]. Synthetic databases had combinations of different numbers of classes and instances and degrees of imbalance.

variables (LVs) L_1 and L_2, respectively, each of which has three observed variables (OVs), which are proxies for disease symptoms, together $X_1 - X_6$.

Since a future patient disease state is unknown and thus cannot be modeled and predicted, we can instead predict the value of a symptom that hints at the disease state using the values of other symptoms. However, if there is no a priori information that a specific symptom is the most predictive of the disease and we want to eliminate uncertainty, we might repeat and predict each symptom in turn and average the prediction performance over all predicted symptoms. While this strategy may sound very practical, it can neither discover cause–disease relations nor their evolution over time in the domain and thereby cannot contribute to understanding the disease or its progression, assist in drug development, or enable a better cure for the disease.

Therefore, for causal discovery and cause–disease relations monitoring, we may wish to use graphical models such as dynamic BNs and, especially, latent variable models (LVMs) [2, 44, 90, 91, 92]. To meet this challenge, we propose learning a causal latent model in each time slot locally. Then, we suggest local-to-global learning over time slices, based on probabilistic scoring and temporal reasoning to transfer the local graphs into a non-stationary latent dynamic BN (DBN) with intra- and inter-slice edges showing causal interrelationships among latent variables and between latent variables and observed variables. This is performed based on the learning pairwise cluster comparison (LPCC) algorithm [91, 92] using the LPCC-based local-to-global (LGL) algorithm [93] to learn a temporal LVM.

The LPCC-based LGL algorithm was evaluated on data sets that were sampled from the artificial temporal BN in Fig. 6a for varying sequence sizes, $4 \leq T \leq 15$, i.e., a record is $(|\mathbf{O}| \times T)$-dimensional, and $\mathbf{O}$ is the set of observed variables. The sample size was $D = \{2{,}000, 3{,}000, 4{,}000, 5{,}000, 10{,}000\}$. The cardinality of all variables was set to four, where the probability that an observed variable takes the same value as does its parent latent variable is 0.8, and the probability that it takes any other value is equally distributed (i.e., a 0.2 "noise" level was evenly distributed among the other values). These probabilities are the same for all T values to guarantee stationarity. Reported results are averaged over ten data permutations for each value combination of T and D.

We empirically compared the LPCC-based LGL algorithm with the state-of-the-art structural expectation maximization (SEM) algorithm [94, 95] that learns a latent DBN, i.e., the SEM-DBN algorithm. This algorithm uses an S&S procedure to find the best fitted model from data, although not necessarily a causal one. It also requires the user to specify the number of LVs and their cardinalities beforehand. For fairness, we limited it to: (1) search over the (smaller) space of pure measurement models (PMMs),[10] and even initialized it with a random PMM, and (2) not to direct an edge from t to $t - 1$.

[10] A DAG over sets of observed variables, latent variables, and edges is a measurement model if a latent variable is a parent of at least one observed variable, an observed variable is a child of at least one latent variable, and none of the observed variables is a parent of any latent variable. A measurement model is called a pure measurement model (PMM) if each observed variable has a single parent and that parent is a latent variable [90].

For comparison, assuming the absence of an LVM, we also compared the classification accuracy of the LPCC-based LGL algorithm and that of the SEM-DBN with the average accuracy of six RF classifiers, each classifying each of the six OVs of the graph in Fig. 6a in the Tth (last) slice (acting as the classification node, where all the remaining OVs in all slices are predictors). By taking the average over the six classifiers, we avoid any preference in the classification (as above). That is, we compared this straightforward classification approach (denoted as a non-LVM) to that based on identifying latent variables and their values using either the LPCC-based LGL or SEM-DBN algorithms, and using the values of the learned latent variables to perform the classification. Thereby, we plan to demonstrate the contribution of learning a temporal LVM in a classification task compared with ignoring the existence of latent variables. We repeated this comparison for each combination of sequence size, sample size, and data permutation.

In total, we trained $5 \times 8 \times 10 \times 6 \times 3 = 7{,}200$ classifiers for five sample sizes, eight sequence sizes, ten data permutations, six classifiers (classification nodes), and three models (LGL/SEM-DBN/no-LVM). We used 80% of the instances of each data set for training and 20% for testing. Figure 6b shows the accuracy and F1-measure averaged over all data permutations, sequence sizes, and observed variables for different sample sizes of the three models. It shows that the LPCC-based LGL algorithm achieves the best classification performance (significantly superior to the others). It reaches the highest possible accuracy of 80% since the noise was originally set at 20%, i.e., even if the algorithm learns the true classification rule, in 20% of cases it will be wrong. This experiment not only allows us to appreciate the importance of learning an LVM in general but also specifically in classification tasks, since latent mechanisms always exist.

Finally, the LPCC-based LGL algorithm was applied to the ALS open-source *PROACT ALS data set* [86] that consists of 3,171 patients with 22,089 clinic visits, from which we derived a subset of 2,590 patients who had at least four visits, each consecutive two are up to six months apart. A visit consists of lab test results and clinical variables describing patient physical functioning, e.g., in walking, writing, and speech. The LGL-based LPCC learned a graph using four latent variables (Fig. 7) demonstrating patient functioning: bulbar functionality ($L1$) indicated mainly by speech (Sp), salivation (Sv), and swallowing (Sw); gross-motor functionality ($L2$) indicated by walking (Wa) and climbing stairs (Cs); fine-motor functionality ($L4$) indicated by dressing (Dr), writing (Wr), and cutting food (Cu); and full body functionality ($L3$) indicated by turning (Tr) in bed, respiratory (Re) ability, FVC, and two lab tests: CK and chloride (Ch) (also found correlated in [46, 96]). The three intra-slice edges represent the natural connections between the bulbar and gross-motor, gross-motor and full body, and fine-motor and full body functionalities. The inter-slice edges between bulbar and full body to themselves complete the temporal–causal reasoning that resembles medical categorization and convention.

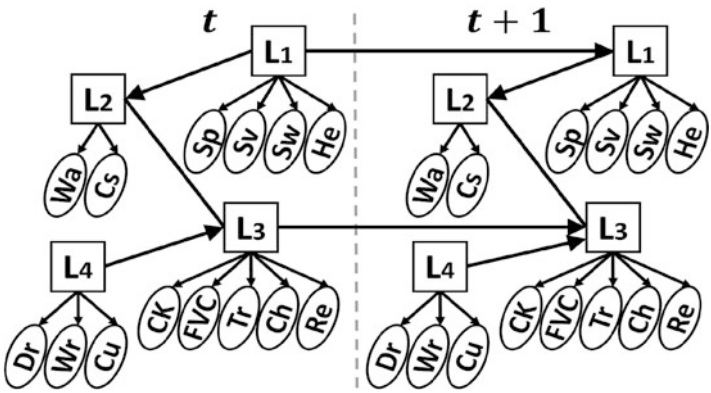


Fig. 7 A temporal LVM learned by the LPCC-based LGL for the ALS data set

5 Conclusion and Discussion

Machine learning models are usually not used in an isolated way but are embedded in some process or product and interact with people. Thus, a more flexible yet holistic view of the entire process, from data collection to the final consumption of the explained prediction, is needed. This includes considering both how to explain predictions to individuals with diverse knowledge and backgrounds and the need for interpretability on the level of an institution or society in general [28, 32]. This is especially required when moving from sandbox studies of benchmark data sets to actual real-world problems [97, 98].

Researchers and practitioners seek to make their algorithms more understandable by focusing, for example, on explicit explanation of decisions and actions to a human observer. However, this focus should range beyond the ML researchers' intuition of what constitutes a "good" explanation and build on existing research from philosophy, cognitive psychology/science, and social psychology, disciplines that grapple with these topics, and study how people define, generate, select, evaluate, and present explanations [37].

This chapter encourages more exploration and exploitation of human-understandable causal models, such as the BNCs, of the operation of ML and especially DNN paradigms. BNC models will allow better introduction and use of causal discovery, interventions, and queries as effective tools to promote explainability and interpretability in developing and applying ML. They have natural interpretability (ability to understand the results) and explainability (ability to explain the results). Unlike ensemble classifiers, the BNC does not provide a list of "important" variables ranked by a statistical, discriminability, or information measure (where the variable "importance" value can differ even by a fraction of a percent and/or due to calibration issues), but also a feature-selection mechanism through variables' Markov blankets and hierarchies of connections along with paths

of influence and inference mechanisms that together expose causal relations in the domain.

A main challenge is that the BN/BNC is not suitable for processing high-dimensional and/or non-semantic data that exist in domains based, for example, on images, text, and speech. Current BN learning algorithms are limited to small to medium domains usually of discrete variables. While it is hard to believe that this will be changed in the near future, it is more reasonable to expect BNs to be applied to processed, projected, or embedded DNN representations of high-dimensional domain input. DNN projections and embeddings of high-dimensional data already reflect semantic evidence that can more easily be extracted, analyzed, and exploited by the BN compared, for example, to the same data re-processed or re-embedded by more layers of the DNN. Effective BN-based tools to promote explainability and interpretability can, for example, allow the user to understand the chain of causal effects from DNN input, to low-level features of the domain, to high-level human-understandable concepts, to DNN outputs [99]. Such a capability is a powerful tool for debugging, understanding bias, and ensuring the safe operation of AI systems. Additionally, these tools may extract low-dimensional concepts from DNNs to generate a human-understandable "vocabulary" and learn a BN that relates the DNN's inputs to the concepts, and the concepts to the DNN's outputs [43]. Other probabilistic models, such as the hidden Markov model (for multilayer perceptron networks) and Gaussian mixture model (for convolutional neural networks), may extract activity patterns of the network hidden layers, where transition probabilities between clusters (mixture components) in consecutive modeled layers may be estimated from the data [42]. Then nodes and paths relevant for network prediction can be chosen, connected, and visualized as an inference graph. This graph is useful for understanding the general inference process for a class, as well as explaining decisions the network makes regarding specific images. Also, a scalable graphical-model framework [100] was shown to aid human understanding and reasoning by providing criticism to explain what is not captured by the examples and improving the interpretability of complex data distributions. In addition, it was demonstrated [101] that when casting the problem of learning the connectivity of a DNN as a BN structure learning problem according to the recursive autonomy identification notion [27], the resulting DNN structure encodes independencies in the input distribution hierarchically, where lower-order independencies are encoded in deeper layers. Tools and mechanisms such as these may also facilitate the struggle of DNN researchers to interpret hidden layer activity in order to understand the general inference process of a classifier, explain decisions the network makes, and indicate the relevance of specific inputs to the network output.

To conclude, returning to Footnote 4, we believe that transparency and accountability are hard to achieve by BOTH human and machine decision-making, and thus we call for more human involvement and intervention for confirmation and validation in AI-driven systems. This could be accomplished by the development of graphical user interface tools soliciting, fostering, and supporting human–machine interaction and bi-directional communication by which, on the one hand, users' inquiries will manipulate and extend the learned BNC model to better address these

and further inquiries, and, on the other hand, the tools will inspire users' curiosity to further interrogate and explore the model to enrich their understanding of the domain beyond what they expected to achieve when running the tool. Bayesian network classifiers can confer transparency on DNNs and other ML tools, enable interpretability and explainability, and empower humans—ML experts and non-ML users alike—to understand but also to affect the results of these tools, all of which will increase trust and trustworthiness in ML to deliver true "explainable AI." To maximize the tremendous impact that ML is having on all aspects of our lives, and enhance trustworthiness in AI, let us embrace the opportunities BNCs are bringing us.

Acknowledgements The author thanks Nuaman Asbeh, Idit Bernstein, Jonathan Gordon, Yaniv Gurwicz, Dan Halbersberg, Roy Kelner, Yael Konforti, Lev Koushnir, Roy Malka, Yair Meidan, Alon Shpigler, Maydan Wienreb, and Raanan Yehezkel (Rohekar) for the use of their dissertations' results in this chapter, and Aviad Raz and Aharon Bar-Hillel for comments and feedback that helped improve this chapter.

References

1. Pearl, J.: Probabilistic reasoning in intelligent systems: Networks of plausible inference. Morgan Kaufmann, San Francisco (1988)
2. Spirtes, P., Glymour, C., Scheines, R.: Causality, prediction and search (2nd edition). MIT Press, Cambridge, MA (2000)
3. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society, Series B, **50**, 157–224 (1988)
4. Cooper, G.F., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Machine Learning, **9**, 309–347 (1992)
5. Heckerman,D.: A tutorial on learning with Bayesian networks. Microsoft Research Technical Report MSR-TR-95-06 March 1995 (revised November 1996)
6. Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: the combination of knowledge and statistical data. Machine Learning, **20**, 197–243 (1995)
7. Meek, C.: Strong completeness and faithfulness in Bayesian networks. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, 411–418 (1995)
8. Chickering, D.M.: Optimal structure identification with greedy search. Journal of Machine Learning Research, **3**, 507–554 (2002)
9. Friedman, N., Geiger, D., Goldszmidt, M: Bayesian network classifiers. Machine Learning, **29**, 131–163 (1997)
10. Kontkanen, P., Myllymaki, P., Sliander, T., Tirri, H.: On supervised selection of Bayesian networks. Proceedings of the Fifteenth International Conference on Uncertainty in Artificial Intelligence, 334–342 (1999)
11. Grossman, D., Domingos, P.: Learning Bayesian network classifiers by maximizing conditional likelihood. Proceedings of the 21st International Conference on Machine Learning, 361–368 (2004)
12. Guo, Y., Greiner, R.: Discriminative model selection for belief net structures. Proceedings of the AAAI, 770–776 (2005)
13. Pernkopf, F.: Bayesian network classifiers versus selective k-NN classifier. Pattern Recognition, **38**, 1–10 (2005)

14. Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., Tirri, H.: On discriminative Bayesian network classifiers and logistic regression. Machine Learning, **59**, 267–296 (2005)
15. Acid, S., Campos, L., Castellano, J.: Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs. Machine Learning, **59**, 213–235 (2005)
16. Pernkopf, F., Bilmes, J.A.: Efficient heuristics for discriminative structure learning of Bayesian network classifiers. Journal of Machine Learning Research, **11**, 2323–2360 (2010)
17. Kelner, R., Lerner, B.: Learning Bayesian network classifiers by risk minimization. International Journal of Approximate Reasoning **53**, 248–272 (2012)
18. Cheng, J., Greiner, R.: Comparing Bayesian network classifiers. Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, 101–108 (1999)
19. Akaike, H.: Information theory and an extension of the maximum likelihood principle. In: Parzen E., Tanabe K., Kitagawa G. (eds) Selected Papers of Hirotugu Akaike. Springer Series in Statistics (Perspectives in Statistics). Springer, New York, NY (1998)
20. Schwarz, G.: Estimating the dimension of a model. The Annals of Statistics, **6**, 461–464 (1978)
21. Kullback, S., Leibler, R.: On information and sufficiency. The Annals of Mathematical Statistics, **22**, 79–86 (1951)
22. Martínez, A.M., Webb, G.I., Chen, S., Zaidi, N.A.: Scalable learning of Bayesian network classifiers. Journal of Machine Learning Research, **17**, 1–35 (2016)
23. Jing, Y., Pavlović, V. Rehg, J.M.: Boosted Bayesian network classifiers. Machine Learning, **73**, 155–184 (2008)
24. Carvalho A.M., Oliveira A.L., Sagot MF.: Efficient learning of Bayesian network classifiers. In: Orgun M.A., Thornton J. (eds) AI 2007: Advances in Artificial Intelligence. AI 2007. Lecture Notes in Computer Science, vol 4830. Springer, Berlin, Heidelberg (2007)
25. Madden, M.G.: On the classification performance of TAN and general Bayesian networks. Knowledge-Based Systems, **22**, 489–495 (2009)
26. Keogh, E., Pazzani, M.: Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. Proceedings of the International Workshop on Artificial Intelligence and Statistics, pp. 225–230 (1999)
27. Yehezkel, R., Lerner, B.: Bayesian network structure learning by recursive autonomy identification. Journal of Machine Learning Research, **10**, 1527–1570 (2009)
28. Molnar, C., Casalicchio, G., Bischl, B.: Interpretable machine learning–A brief history, state-of-the-art and challenges. arXiv:2010.09337 (2020)
29. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144 (2016)
30. Molnar, C., König, G., Herbinger, J., Freiesleben, T., Dandl, S., Scholbeck, C.A., Casalicchio, G., Grosse-Wentrup, M., Bischl, B.: Pitfalls to avoid when interpreting machine learning models. ICML 2020 Workshop XXAI: Extending Explainable AI Beyond Deep Models and Classifiers (2020)
31. Lo Piano, S.: Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward. Humanities and Social Sciences Communications, **7**, 9 (2020)
32. Spiegelhalter, D.: Should we trust algorithms? Harvard Data Science Review, 2(1) (2020)
33. Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., et al.: Toward trustworthy AI development: Mechanisms for supporting verifiable claims. arXiv e-print arXiv:2004.07213 (2020)
34. Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., Zhong, C.: Interpretable machine learning: Fundamental principles and 10 grand challenges, **CoRR**, abs/2103.11251, https://arxiv.org/abs/2103.11251 (2021)
35. Kroll, J.A., Huey, J., Barocas, S., Felten, E.W., Reidenberg, J.R., Robinson, D.G., Yu, H.: Accountable algorithms, 165 University of Pennsylvania Law Review 633 (2017)
36. Ashoori, M., Weisz, J.D.: In AI we trust? Factors that influence trustworthiness of AI-infused decision-making processes. arXiv e-print arXiv:1912.02675 (2019)

37. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence, **267**, 1–38 (2019)
38. Lundberg, S.M., Lee, S-I.: A unified approach to interpreting model predictions. In: Guyon, I., et al. (eds) Proceedings of the Advances in Neural Information Processing Systems 30, 4765–4774 (2017)
39. Olah, C., Mordvintsev, A., Schubert, L.: Feature visualization. Distill, **2** (2017)
40. Selvaraju, R.R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., Batra, D.: Grad-CAM: Why did you say that? Visual explanations from deep networks via gradient-based localization, CoRR, **abs/1610.02391**, http://arxiv.org/abs/1610.02391 (2016)
41. Binder A., Bach S., Montavon G., Müller KR., Samek W.: Layer-wise relevance propagation for deep neural network architectures. In: Kim K., Joukov N. (eds) Information Science and Applications (ICISA). Lecture Notes in Electrical Engineering, 376. Springer, Singapore (2016)
42. Konforti, Y., Shpigler, A., Lerner, B., Bar Hillel, A.: SIGN: Statistical inference graphs based on probabilistic network activity interpretation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **45**(3), 3783–3797 (2023)
43. Harradon, M., Druce, J., Ruttenberg, B.: Causal learning and explanation of deep neural networks via autoencoded activations, arXiv:1802.00541 (2018)
44. Pearl, J.: Causality: Models, reasoning, and inference (2nd edition), Cambridge University Press, New York, NY (2009)
45. Peters, J., Janzing, D., Schölkopf, B.: Elements of causal inference - Foundation and learning algorithms. The MIT Press (2017)
46. Gordon, J., Lerner, B.: Insights into ALS from a machine learning perspective. Journal of Clinical Medicine, **8**, 1578 (2019)
47. Drugan, M.M., Wiering, M.A.: Feature selection for Bayesian network classifiers using the MDL-FS score. International Journal of Approximate Reasoning, **51**, 695–717 (2010)
48. dos Santos, E.B., Hruschka Jr., E.R., Hruschka, E.R., Ebecken, N.F.F.: Bayesian network classifiers: Beyond classification accuracy. Intelligent Data Analysis, **15**, 279–298 (2011)
49. Bielza, C., Larrañaga, P.: Discrete Bayesian network classifiers: A survey. ACM Computing Surveys, **47** 1–43 (2014)
50. Aliferis, C.F., Tsamardinos, I., Statnikov, A.: HITON: a novel Markov blanket algorithm for optimal variable selection. AMIA Annual Symposium Proceedings, 21–25 (2003)
51. Tan, Y., Liu, Z.: Feature selection and prediction with a Markov blanket structure learning algorithm. BMC Bioinformatics, **14** (Suppl 17), A3 (2013)
52. Antal, P., Millinghoffer, A., Hullám, G., Szalai, C,. Falus, A.: A Bayesian view of challenges in feature selection: Feature aggregation, multiple targets, redundancy and interaction. In: Saeys, Y., Liu, H., Inza, I., Wehenkel, L., Pee, Y. (eds) Proceedings of the Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery at ECML/PKDD 2008, Proceedings of Machine Learning Research, 74–89 (2008)
53. Shih, A., Choi, A., Darwiche, A.: A Symbolic approach to explaining Bayesian network classifiers. Proceedings of the 27th International Joint Conference on Artificial Intelligence, 5103–5111 (2018)
54. Langley, P., Iba, W., Thompson, K.: An analysis of Bayesian classifiers. Proceedings of the Tenth National Conference on Artificial Intelligence, 223–228 (1992)
55. Lerner, B.: Bayesian fluorescence in situ hybridisation signal classification. Artificial Intelligence in Medicine, **30**, 301–316 (2004)
56. Lerner, B., Lawrence N. D.: A comparison of state-of-the-art classification techniques with application to cytogenetics. Neural Computing & Applications, **10**, 39–47 (2001).
57. Lerner, B., Yeshaya, J., Koushnir, L.: On the classification of a small imbalanced cytogenetic image database. IEEE-ACM Transactions on Computational Biology and Bioinformatics, **4**, 204–215 (2007)
58. Lerner, B., Koushnir, L., Yeshaya, J.: Segmentation and classification of dot and non-dot-like fluorescence in-situ hybridization signals for automated detection of cytogenetic numerical abnormalities. IEEE Transactions on Information Technology in Biomedicine, **11**, 443–449 (2007)

59. Webb, G.I., Boughton, J.R., Wang, Z.: Not so naive Bayes: Aggregating one dependence estimators. Machine Learning, **58**, 5–24 (2005)
60. Zheng, Z., Webb, G.I.: Lazy learning of Bayesian rules. Machine Learning, **41**, 53–84 (2000)
61. Xie, Z., Hsu, W., Liu, Z., Lee, M.: A selective neighborhood based naive Bayes for lazy learning. In: Chen, M.-S., Yu, P.S., Liu, B. (eds.) PAKDD 2002. LNCS (LNAI), **2336**, 104–114. Springer, Heidelberg (2002)
62. Frank, E., Hall, M., Pfahringer, B.: Locally weighted naive Bayes. Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), 249–256. Morgan Kaufmann Publishers, Seattle (2003)
63. Chan, H., Darwiche, A.: Reasoning about Bayesian network classifiers, **CoRR**, abs/1212.2470, http://arxiv.org/abs/1212.2470 (2012)
64. Meidan, Y., Lerner, B., Rabinowitz, G., Hassoun, M.: Cycle-time key factor identification and prediction in semiconductor manufacturing using machine learning and data mining. IEEE Transactions on Semiconductor Manufacturing, **24**, 237–248 (2011)
65. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory, **14**, 462–467 (1968)
66. Lerner, B., Malka, R.: Investigation of the K2 algorithm in learning Bayesian network classifiers. Applied Artificial Intelligence, **25**, 74–96 (2011)
67. Geiger, D., Heckerman, D.: Knowledge representation and inference in similarity networks and Bayesian multinets. Artificial Intelligence, **82**, 45–74 (1996)
68. Bilmes, J.: Dynamic Bayesian multinets. Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI), Morgan Kaufmann Publishers (2000)
69. Gurwicz, Y., Lerner, B.: Bayesian class-matched multinet classifier. SSPR/SPR, ser. Lecture Notes in Computer Science, D. Y. Yeung, J. T. Kwok, A. L. N. Fred, F. Roli, and D. de Ridder, Eds., Springer, Vol. 4109, 145–153 (2006)
70. Pena, J.M., Lozano, J.A., Larranaga, P.: Learning recursive Bayesian multinets for data clustering by means of constructive induction. Machine Learning, **47**, 63–89 (2002)
71. Dheeru, D., Casey, G.: UCI machine learning repository, http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences (2017)
72. Yang, Y., Korb, K., Ting, K.M., Webb, G.I.: Ensemble selection for SuperParent-One-Dependence estimators. In Lecture Notes in Computer Science: Proceedings of the 18th Australian Conference on AI (AI 05), volume LNCS 3809, pages 102–111. Berlin: Springer (2005)
73. Jiang, L., Zhang, H.: Lazy averaged one-dependence estimators. In: Lamontagne, L., Marchand, M. (eds.) Advances in Artificial Intelligence, Canadian AI 2006. Lecture Notes in Computer Science, **4013**. Springer, Berlin, Heidelberg (2006)
74. Lam, W., Bacchus, F.: Learning Bayesian belief networks: an approach based on the MDL principle. Computational Intelligence, **10**, 269–293 (1994)
75. Dawid, A.P.: Present position and potential developments: Some personal views. Statistical theory. The prequential approach. Journal of Royal Statistical Society, Series A, **147**, 278–292 (1984)
76. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. International Joint Conference on Artificial Intelligence, 1137–1143 (1995)
77. Vapnik, V.N.: Statistical learning theory. John Wiley & Sons, New York (1998)
78. Cowell, R.: Introduction to inference for Bayesian networks. In: M. I. Jordan (ed.) Learning Graphical Models, 9–26. MIT Press, Cambridge, Massachusetts (1999)
79. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.H.: Classification and Regression Trees, Wadsworth, Belmont, CA (1984)
80. Breiman, L.: Random forests. Machine Learning, **45**, 5–32 (2001)
81. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research, **7**, 1–30 (2006)
82. Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press, UK (1995)
83. Cortes, C., Vapnik, V.: Support vector networks. Machine Learning, **20**, 273–297 (1995)

84. Macià, N., Bernadó-Mansilla, E.: Towards UCI+: A mindful repository design. Information Sciences, **261**, 237–262 (2014)
85. Malka, R., Lerner, B.: Classification of fluorescence in-situ hybridization images using belief networks. Pattern Recognition Letters, **25**, 1777–1785 (2004)
86. Atassi, N., Berry, J., Shui, A., Zach, N., Sherman, A., Sinani, E., et al.: The PRO-ACT database design, initial analyses, and predictive features. Neurology, **83**, 1719–1725 (2014)
87. Halbersberg, D., Lerner, B.: Young driver fatal motorcycle accident analysis by jointly maximizing accuracy and information. Accident Analysis and Prevention, **129**, 350–361 (2019)
88. Halbersberg, D., Wienreb, M., Lerner, B.: Joint maximization of accuracy and information for learning the structure of a Bayesian network classifier. Machine Learning, **109**, 1039–1099 (2020)
89. Halbersberg, D., Lerner, B.: Learning a Bayesian network classifier by jointly maximizing accuracy and information, Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI), The Hague, Holland, 1638–1639 (2016)
90. Silva, R., Scheines, R., Clark, G., Spirtes, P.: Learning the structure of linear latent variable models. Journal of Machine Learning Research, **7**, 191–246 (2006)
91. Asbeh, N., Lerner, B.: Learning latent variable models by pairwise cluster comparison. Part I - Theory and overview. Journal of Machine Learning Research, **17** (224), 1–52 (2016)
92. Asbeh, N., Lerner, B.: Learning latent variable models by pairwise cluster comparison. Part II - Algorithm and evaluation. Journal of Machine Learning Research, **17** (233), 1–45 (2016)
93. Halbersberg, D., Lerner, B.: Local to global learning of a latent dynamic Bayesian network. In: De Giacomo, G., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang. J. (eds.) ECAI 2020. Proceedings of the 24th European Conference on Artificial Intelligence, Santiago de Compostela, Spain, Frontiers in Artificial Intelligence and Applications, **325**, 2600–2607, IOS Press (2020)
94. Friedman, N.: The Bayesian structural EM algorithm. Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 129–138, Morgan Kaufmann Publishers Inc. (1998)
95. Murphy, K.P.: Machine learning: A probabilistic perspective. MIT Press (2014)
96. Ikeda, K., Hirayama, T., Takazawa, T., Kawabe, K., Iwasaki, Y.: Relationships between disease progression and serum levels of lipid, urate, creatinine and ferritin in Japanese patients with amyotrophic lateral sclerosis: a cross-sectional study. Internal Medicine, **51**, 1501–1508 (2012)
97. Wagstaff, K.L.: Machine learning that matters. In Proceedings of International Conference on Machine Learning (ICML), 529–536 (2012)
98. Rudin, C. Wagstaff, K.L.: Machine learning for science and society. Machine Learning, **95**, 1–9 (2014)
99. Woodward, J.: Making things happen: A theory of causal explanation. Oxford University Press (2005)
100. Kim, B., Khanna, R., Koyejo, O.: Examples are not enough, learn to criticize! Criticism for interpretability. In: D. Lee and M. Sugiyama and U. Luxburg and I. Guyon and R. Garnett (eds.), Advances in Neural Information Processing Systems, **29**, Curran Associates, Inc. (2016)
101. Rohekar, R.Y., Nisimov, S., Gurwicz, Y., Koren, G., Novik, G.: Constructing deep neural networks by Bayesian network structure learning. Proceedings of the 32nd International Conference on Neural Information Processing Systems, 3051–3062, Montréal, Canada (2018)

Soft Decision Trees

Oren Fivel, Moshe Klein, and Oded Maimon

1 Introduction

In this chapter, we develop the foundation of a new theory for decision trees based on new modeling of phenomena with soft numbers. This calls for major concept change of probability, which is developed in this chapter, so that decision trees can be modeled. Soft numbers represent the theory of soft logic that addresses the need to combine real processes and cognitive ones in the same framework. At the same time, soft logic develops a new concept of modeling and dealing with uncertainty: the uncertainty of time and space. It is a language that can talk in two reference frames and also suggest a way to combine them.

1.1 Research Motivation and Direction

Probability theory is used in order to model processes and phenomena, involving randomness of the parameters and variables. A probability of a continuous random variable is defined by a probability density function (PDF). The PDF can be used to approximate the probability of the continuous random variable X to be adjacent to x in the following sense:

O. Fivel (✉)
School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Be'er Sheva, Israel
e-mail: fivel@post.bgu.ac.il

M. Klein · O. Maimon
Department of Industrial Engineering, Tel-Aviv University, Tel Aviv, Israel
e-mail: mosheklein@mail.tau.ac.il; maimon@tauex.tau.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_8

$$\Pr(x < X \leq x + \Delta x) \approx f_X(x)\Delta x, \tag{1}$$

where $\Delta x > 0$ is a small value that defines how much this probability is accurate. However, continuous random variables have the following properties:

- No distinguishing between strict inequality and non-strict in equality, e.g., $\Pr(X \leq x) = \Pr(X < x)$.
- Equality collapses to zero, i.e., $\Pr(X = x) = 0$. Although any value of $x \in S_X$ (S_X denotes the support of X) is possible for X, the probability of X to be equal to any value of $x \in S_X$ is (almost surely) zero.

Because of these properties, we lose some information regarding to a continuous random variable to have an exact value. On one hand, an event "$X = x$" might be possible (if $x \in S_X$) but improbable (i.e., with zero probability), which seems to be a paradox. On the other hand, we can express the zero probability by of an event "$X = x$" by letting Δx to approach to zero in Eq. (1)

$$\Pr(X = x) = f_X(x) \cdot 0. \tag{2}$$

This equation presents the probability $\Pr(X = x)$ as a multiple of zero with a factor of the PDF $f_X(x)$ for all x. Instead of taking $\Pr(X = x)$ to be completely zero, we can assign to it a zero multiple of $f_X(x)$ and compare different probability values for different observation values x. This approach can be implemented by using *soft numbers* (see Appendix 2 and Klein and Maimon's papers, e.g., [5, 6] and [7]).

In addition, there is an approach to represent a discrete distribution as a continuous distribution by a linear combination of Dirac delta functions $\delta(x - x_i)$, or by any approximations of Dirac delta functions, e.g., Gaussian functions (also known as *Gaussian mixture model* or GMM) or rectangular functions (based on a *"Uniformly" Mixture Model* or UMM), etc. (see Eq. (14) for more details). Our approach is to establish the opposite in some sense, i.e., to represent a continuous random variable with a possibility to have discrete values with probability that will not collapse absolutely to zero.

In this chapter, we introduce the *soft numbers* to give a probability interpretation of a continuous random variable to have an exact value that provides distinguishing between strict inequality and non-strict in equality in the probability function. This probability interpretation is implemented by "Soft Probability" [3].

1.2 Organization of the Work

Section 2 incorporates soft numbers into probability theory to present the notion of "Soft Probability." Section 3 presents an example for application on decision-trees-based C4.5 algorithm. Conclusions and suggestion for future research are shown in Sects. 4 and 5, respectively, to summarize this chapter. For completion, Appendix 1

provides an extension of soft probability for complements, union, intersection, and conditional probability. Appendix 2 provides a presentation of soft numbers.

2 Soft Probability: Incorporation of Soft Number into Probability Theory

In order to incorporate the notion of Eq. (72) in Appendix 2, we define a cumulative distribution function (CDF) of a continuous random variable

$$\mathrm{Ps}(X \leq x) = F_X(1 \cdot \bar{0}\dot{+}x), \tag{3}$$

where $\mathrm{Ps}(\cdot)$ is a suggested type of a probability function, dented as a "Soft Probability" [3] instead of a regular probability notation "$\Pr(\cdot)$" or $P(\cdot)$, and $F_X(\cdot)$ is the regular CDF function of the random variable X, but it is applied on a soft number $1 \cdot \bar{0}\dot{+}x$. Our motivation is to generate an alternative evaluation of the probability at the left-hand side (LHS), so that we can distinguish between $\mathrm{Ps}(X < x)$ and $\mathrm{Ps}(X \leq x)$ for a continuous random variable X (i.e., $\mathrm{Ps}(X < x) \neq \mathrm{Ps}(X \leq x)$). We will show that the evaluation of the soft number at the CDF in the right-hand side (RHS) will create this distinction.

The RHS of Eq. (3) can be decomposed by Eq. (72) as follows:

$$F_X(1 \cdot \bar{0}\dot{+}x) \stackrel{\text{def}}{=} f_X(x)\bar{0}\dot{+}F_X(x). \tag{4}$$

The LHS of Eq. (3) can be decomposed by separating the event "$X \leq x''$ into a disjoint union "$X = x \uplus X < x.''$ In a regular probability, we have the known identities

$$\begin{aligned}\Pr(X \leq x) &\stackrel{\text{“}X=x''\cap\text{“}X<x''=\emptyset}{=} \underbrace{\Pr(X = x)}_{=0} + \Pr(X < x) \\ &= \Pr(X < x).\end{aligned}$$

So we do not have a distinction between $\Pr(X \leq x)$ and $\Pr(X < x)$. We distinguish between $\mathrm{Ps}(X \leq x)$ and $\mathrm{Ps}(X < x)$ by the following definition for $\mathrm{Ps}(X \leq x)$

$$\mathrm{Ps}(X \leq x) \stackrel{\text{def}}{=} \mathrm{Ps}(X = x) + \mathrm{Ps}(X < x), \tag{5}$$

so that we define the terms on the LHS as follows:

$$\mathrm{Ps}(X = x) \stackrel{\text{def}}{=} f_X(x)\bar{0}, \tag{6}$$

$$\mathrm{Ps}(X < x) \stackrel{\text{def}}{=} F_X(x) \equiv \Pr(X < x). \tag{7}$$

By this setup, we achieve a distinguishing between $\mathrm{Ps}(X \leq x)$ and $\mathrm{Ps}(X < x)$, and also we provide an interpretation to $\mathrm{Ps}(X = x)$ be infinitesimally small but not collapse completely to zero due to the factor $\bar{0}$ of the PDF.

In the next subsection, we provide two examples of implementation on mixture models PDFs, GMM, and UMM, in order to demonstrate the effect of soft numbers (and more precisely, soft zeros) on PDFs.

2.1 Examples on Mixture Models

In order to demonstrate the effect of soft numbers (and more precisely, soft zeros) on PDFs, we provide two examples of implementation on mixture models:

- Gaussian mixture model (GMM)
- Uniformly mixture model (UMM)

A PDF of a continuous random variable X with mixture model is given by

$$f_X\left(x; \{(\theta_i, w_i)\}_{i=1}^n\right) = \sum_{i=1}^{n} w_i f_i(x; \theta_i), \tag{8}$$

where $f_i(\cdot; \theta_i)$ is a PDF with a set of parameters θ_i, and $w_i \in [0, 1]$ is a weight that multiplies the ith PDF f_i and sums to 1, i.e., $\sum_{i=1}^{n} w_i = 1$. In the GMM case, the ith PDF f_i is parameterized with mean μ_i and variance σ_i^2 such that

$$f_i(x; \mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2\sigma_i^2}(x-\mu_i)^2}. \tag{9}$$

In the UMM case, the ith PDF f_i is parameterized with the open interval (a_i, b_i) such that

$$f_i(x; a_i, b_i) = \frac{1}{b_i - a_i} \mathbb{1}_{x \in (a_i, b_i)}, \tag{10}$$

where $\mathbb{1}_A$ is the indication function that indicates "1" if "A" is true and "0" if "A" is false. The corresponding soft probability is obtained by

$$\mathrm{Ps}\left(X = x; \{(\theta_i, w_i)\}_{i=1}^n\right) = \sum_{i=1}^{n} w_i f_i(x; \theta_i) \cdot \bar{0}. \tag{11}$$

For illustration, we chose following parameters for the mixture model with 2 components each and plot them in Fig. 1:

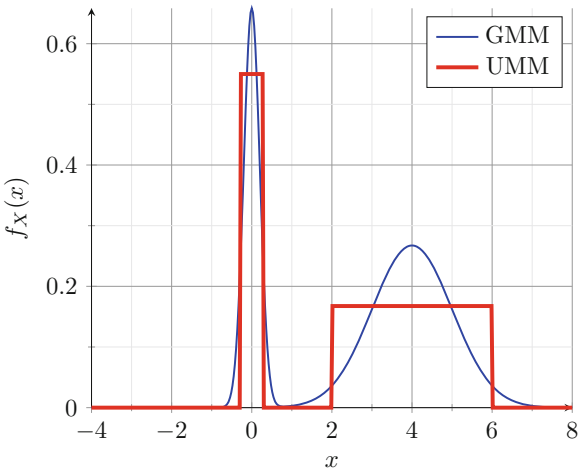


Fig. 1 Examples of Gaussian mixture model and uniformly mixture model's PDFs

- GMM: $(\mu_1, \sigma_1, w_1) = (0, 0.2, 0.33)$, $(\mu_2, \sigma_2, w_2) = (4, 1, 0.67)$.
- UMM: $((a_1, b_1), w_1) = ((-0.3, 0.3), 0.33)$, $((a_2, b_2), w_2) = ((2, 6), 0.67)$.

In the GMM case, we have two local maximums at $x = \mu_1$ (a global maximum) and $x = \mu_2$. Hence, if we take three points $x_1 \approx \mu_1$, $x_2 \approx \mu_2$, and x_3 such that $|x_1|, |x_2| \ll |x_3|$, we have the following order of soft probabilities:

$$\text{Ps}(X = x_1) > \text{Ps}(X = x_2) > \text{Ps}(X = x_3) > 0 \cdot \bar{0}.$$

We have a strict inequality in the RHS because the support of the GMM is infinite. The soft probability of the GMM presents an absolute low probability of X to have an exact value x but relative high probability when X is closer to μ_1 (where the GMM is maximal in this case). In other words, in contrast to classic probability that would collapse absolutely to have exact value $X = x$, with soft probability, we are able to distinguish (and order) among events $X = x$ that will have a positive soft probability (due to infinite support in GMM).

In the UMM case, if we take three points $x_1 \in (a_1, b_1)$ [global maximum], $x_2 \in (a_2, b_2)$, and $x_3 \notin (a_1, b_1) \cup (a_2, b_2)$, we have the following order of soft probabilities:

$$\text{Ps}(X = x_1) > \text{Ps}(X = x_2) > \text{Ps}(X = x_3) = 0 \cdot \bar{0}.$$

We have equality to absolute zero in the RHS because x_3 is outside of the support $(a_1, b_1) \cup (a_b, b_2)$, while for $i = 1, 2$ we assign some probability values by the soft zero $\text{Ps}(X = x_i) = \frac{w_i}{b_i - a_i} \cdot \bar{0} > 0 \cdot \bar{0}$. Hence, with soft probability, we succeed to distinguish between "zero probabilities" of "impossible" events ($X = x$ with x is outside of the support) and "possible" events ($X = x$ with x is in the support) that would collapse to zero in the classical probability sense, so that the "impossible"

events will be absolute zero and the "possible" events will be some soft zero and ordered accordingly.

2.2 Observations

When it comes to the development of soft numbers, we have two options how to define an absolute value of a soft number: Option 1 is by the definition in Eq. (72) with $|x|' = \text{sign}(x)$, ignoring the fact that this derivative is not continuous, so that

$$|a\bar{0}\dot{+}x| = \alpha\bar{0} \cdot \text{sign}(x)\dot{+}|x|. \tag{12}$$

Option 2 is to define a soft conjugate of $\alpha\bar{0}\dot{+}x$ to be $(-\alpha)\bar{0}\dot{+}x$ such that

$$\begin{aligned} |\alpha\bar{0}\dot{+}x| &= \sqrt{(\alpha\bar{0}\dot{+}x)((-\alpha)\bar{0}\dot{+}x)} \\ &= \sqrt{-(\alpha\bar{0})^2 + x^2} \\ &= \sqrt{-0 + x^2} \\ &= \sqrt{x^2} \\ &= |x|. \end{aligned} \tag{13}$$

If we use Option 2, then we can have the following properties for a soft probability on a continuous random variable:

1. $\text{Ps}(X \leq x) \neq \text{Ps}(X < x)$,
   but $|\text{Ps}(X \leq x)| = |\text{Ps}(X < x)| > |\text{Ps}(X = x)| = 0$.
2. $f_X(x) > f_X(y) \Rightarrow \text{Ps}(X = x) > \text{Ps}(X = y)$,
   but $|\text{Ps}(X = x)| = |\text{Ps}(X = y)| = 0$.
3. $f_X(x) > f_Y(y) \Rightarrow \text{Ps}(X = x) > \text{Ps}(Y = y)$,
   but $|\text{Ps}(X = x)| = |\text{Ps}(Y = y)| = 0$.
4. $|\text{Ps}(X \leq x)| = \text{Ps}(X < x) = \text{Pr}(X < x) = \text{Pr}(X \leq x)$.

By taking absolute values of the soft probability term, we return to the classic probability results for continuous random variable, e.g., not distinguishing between strict inequality and non-strict inequality, and equality collapses to zero.

In the literature (e.g., [8, 4] and [13]), there is an approach to represent a discrete distribution as a continuous distribution by a linear combination of Dirac delta functions $\delta(x - x_i)$, or by any approximations of Dirac delta functions based on a mixture model, e.g., GMM or UMM. Suppose X is a discrete random variable with the probability $\text{Pr}(X = x_i) = p_i$. Then X can be represented with a continuous distribution as follows:

$$
\begin{aligned}
f_X(x) &= \sum_i p_i \delta(x - x_i) \\
&\approx \sum_i p_i \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-x_i)^2}, \sigma^2 \ll 1 \\
&\approx \sum_i p_i \cdot \frac{1}{2a} \mathbb{1}_{x-x_i \in (-a,a)}, a \ll 1.
\end{aligned}
\tag{14}
$$

Recall for Dirac delta function properties,

$$
\begin{aligned}
\delta(x) &= \begin{cases} 0 & x \neq 0 \\ \infty & x = 0 \end{cases} \\
\int_{-\infty}^{\infty} \delta(x) dx &= 1,
\end{aligned}
\tag{15}
$$

and also $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}x^2} \xrightarrow{\sigma^2 \to 0} \delta(x)$, $\frac{1}{2a}\mathbb{1}_{x \in (-a,a)} \xrightarrow{a \to 0} \delta(x)$, i.e., Gaussian distribution and uniformly distribution converge to Dirac delta function (degenerative distribution) when the variance of the Gaussian distribution and the length of the interval in the uniformly distribution approach to zero, respectively (cf. Fig. 1). Our approach is to establish the opposite in some sense, i.e., to represent a continuous random variable with a possibility to have discrete values with probability that will not collapse absolutely to zero. See Appendix 1 for the extension of the soft probability's notion to complement, union intersection, and conditional probability

In this chapter, we introduce the *soft numbers* (see Klein and Maimon's papers, e.g., [5, 6] and [7]) to give a probability interpretation of a continuous random variable to have an exact value that provides distinguishing between strict inequality and non-strict in equality in the probability function.

In the next section, we use the notion of "Soft Probability" into decision trees.

3 Soft Decision Trees

3.1 Overview

Decision trees are simple yet successful techniques for predicting and explaining the relationship between some measurements about an item and its target value (see, e.g., [12, 11]). In most decision trees inducers, discrete splitting functions (also known as *Splitting Criteria*) are univariate, i.e., an internal node is split according to the value of a single attribute. There are various top-down decision tree inducers such as: **ID3** (Ref. [10]), the basic algorithm, **C4.5** (Ref. [9]), an extension of ID3

that can handle continuous variables, **CART** (Ref. [1]), classification and regression tree, etc.

In the next subsection, we introduce an implementation of *decision trees with soft numbers*, based on C4.5 algorithm, in order to generate *Soft Decision Trees*.

3.2 Soft Decision Trees Based on C4.5 Algorithm

In this subsection, we introduce an implementation of *decision trees with soft numbers*, based on C4.5 Algorithm, in order to generate *soft decision trees*, based on a discrete label S and continuous features $X_0, X_1, \ldots, X_n$.

3.2.1 Preferring a Thesis in Equilibrium State

On each stage in the C4.5 algorithm, we search the splitting that will minimize the classes entropy:

$$\min\{\mathrm{H}(S|X_0), \mathrm{H}(S|X_1), \mathrm{H}(S|X_2) \ldots . \mathrm{H}(S|X_n)\}. \tag{16}$$

We will split our data according to the feature that has the most information gain (IG) see e.g., Fig. 2. In some cases when calculating the IG we may find that several features have the same IG, and in the case that the IG is the maximum of all feature's IG we will not be able to tell by which feature we should split our data.

For describing the problem we are trying to solve, we will define the following example:

Let $\{X_0, X_1 \ldots X_n\}$ be continuous features with known distributions $f_{X_i}(x)$, and S be a binary class such that $S \in \{s_1, s_2\}$. For finding the best splitting features, we will calculate the IG for each feature and for the optimum threshold of that feature.

$$\min\{\mathrm{IG}(X_0 = x_{0_{\mathrm{th}}}), \mathrm{IG}(X_1 = x_{1_{\mathrm{th}}}) \ldots \mathrm{IG}(X_n = x_{n_{\mathrm{th}}})\}. \tag{17}$$

In some cases, we can have an equality between two or more IGs of different features, and in that case, there is no preferred thesis, so we can pick one of the suggested features randomly, for example:

$$\mathrm{IG}(X_1 = x_{1_{\mathrm{th}}}) = \mathrm{IG}(X_4 = x_{4_{\mathrm{th}}}) = \mathrm{IG}(X_8 = x_{8_{\mathrm{th}}}).$$

Using the soft probability, we suggest a method to derive the information gain of those features and also the ability to choose the preferred one (Fig. 3). Recall the definition of information gain:

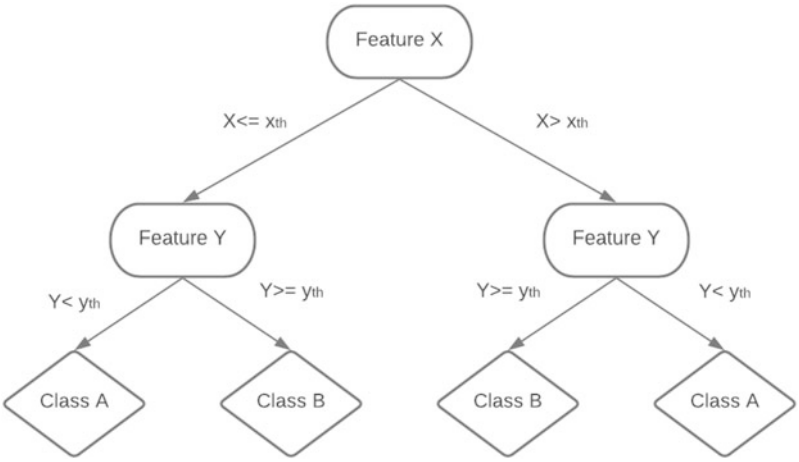


Fig. 2 Classic decision tree splitting

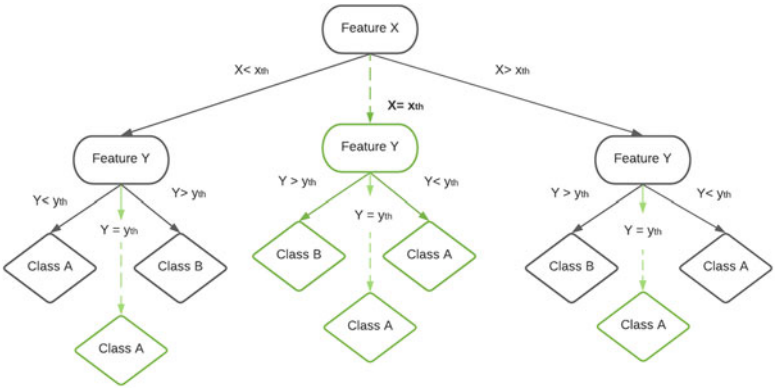


Fig. 3 Soft decision tree splitting

$$\begin{aligned}\mathrm{IG}(X_i) &= \Pr(X_i \geq x_i)\mathrm{H}(S|X_i \geq x_i) + \Pr(X_i < x_i)\mathrm{H}(S|X_i < x_i) \\ &= \Pr(X_i > x_i)\mathrm{H}(S|X_i > x_i) + \Pr(X < x_i)\mathrm{H}(S|X_i < x_i) \\ &\quad + \Pr(\boldsymbol{X_i = x_i})\mathrm{H}(\boldsymbol{S|X_i = x_i}).\end{aligned} \tag{18}$$

The part $\Pr(X_i = x_i)\mathrm{H}(S|X_i = x_i)$ is the IG on the threshold point. It is infinitesimal and treated as 0, so it has no impact on the total IG of each feature. $P(X_i = x) = 0$ in the classical probability sense, so in features equality we get the following in the case of $\mathrm{IG}(X_i) = \mathrm{IG}(X_j)$:

$$\begin{aligned}&\Pr(X_i > x_i)\mathrm{H}(S|X_i \geq x_i) + \Pr(X_i < x_i)\mathrm{H}(S|X_i < x) \\ &\quad = \Pr(X_j > x_j)\mathrm{H}(S|X_j > x_j) + \mathrm{H}(X_j < x_j)\mathrm{H}(S|X_j < x_j).\end{aligned} \tag{19}$$

On the other hand, using soft probability, the part $\mathrm{Ps}(X_i = x)$ does have defined value by Eq. (6), $\mathrm{Ps}(X_i = x) = f_{X_i}(x) \cdot \bar{0}$, so we can express soft information gain as follows:

$$\begin{aligned}\mathrm{IGs}(X_i) &= \mathrm{Ps}(X_i = x_i) \cdot \mathrm{H}(S|X_i = x_i) \\ &\quad + \mathrm{Ps}(X_i > x_i)\mathrm{H}(S|X_i > x_i) + \mathrm{Ps}(X < x_i)\mathrm{H}(S|X_i < x_i) \\ &= f_{X_i}(x_i)\mathrm{Hs}(S|X_i = x_i) \cdot \bar{0} \\ &\quad \dot{+}\Pr(X_i > x_i)\mathrm{H}(S|X_i > x_i) + \Pr(X < x_i)\mathrm{H}(S|X_i < x_i).\end{aligned} \tag{20}$$

When comparing the soft IGs, we realize that we can distinguish among them by comparing the middle branch—threshold branch:

$$f_{X_i}(x_i)\mathrm{H}(S|X_i = x_i) \cdot \bar{0} \overset{>}{\underset{<}{?}} f_{X_j}(x_j)\mathrm{H}(S|X_j = x_j) \cdot \bar{0}. \tag{21}$$

To further realize the subject, we will present a solution to the following problem in the next example.

3.2.2 Example: Uniformly Distribution Features and Discrete Label

Let $X \sim U(0, 2)$ and $Y \sim U(0, 4)$ be features of a data set and $S \in \{0, 1\}$ be an event that we are trying to predict its value according to the features (X, Y). Finding the soft information gain of splitting according to each feature has the following result:

$$\begin{aligned}\mathrm{IG}(X) &= \mathrm{IG}(Y) \\ &= \Pr(X \leq x_{\mathrm{th}})\mathrm{H}(S|X \leq x_{\mathrm{th}}) + \Pr(X > x_{\mathrm{th}})\mathrm{H}(S|X > x_{\mathrm{th}}) \\ &= \Pr(Y \leq y_{\mathrm{th}})\mathrm{H}(S|Y \leq y_{\mathrm{th}}) + \Pr(Y > y_{\mathrm{th}})\mathrm{H}(S|Y > y_{\mathrm{th}}).\end{aligned} \tag{22}$$

With the classic approach, we should choose one of the features X or Y randomly as they both got the same information gain. Using the soft decision approach, we can try to identify which hypothesis is superior even if it is by a small margin, but it may translate to better overall decision-making tool. First we will express the full equation explicitly separating the threshold branch as follows:

$$\begin{aligned}&\Pr(X = x_{\text{th}})\mathrm{H}(S|X = x_{\text{th}}) + [\Pr(X < x_{\text{th}})\mathrm{H}(S|X < x_{\text{th}}) \\ &\quad + \Pr(X > x_{\text{th}})\mathrm{H}(S|X > x_{\text{th}})] \\ &= \Pr(Y = y_{\text{th}})\mathrm{H}(S|Y = y_{\text{th}}) + [\mathrm{Ps}(Y < y_{\text{th}})\mathrm{H}(S|Y < y_{\text{th}}) \\ &\quad + \Pr(Y > y_{\text{th}})\mathrm{H}(S|Y > y_{\text{th}})].\end{aligned} \tag{23}$$

We can say as we have shown before that the last two parts of the equation (in square brackets) are equal and all it is left is to compare the threshold branch:

$$\Pr(X = x_{\text{th}})\mathrm{H}(S|X = x_{\text{th}}) \underset{<}{\overset{>}{=}}? \Pr(Y = y_{\text{th}})\mathrm{H}(S|Y = y_{\text{th}}). \tag{24}$$

In the classical probability, we have equality because $\Pr(X = x_{\text{th}}) = \Pr(Y = y_{\text{th}}) = 0$. In order to compare the threshold branch, we convert Eq. (24) into soft probability (replacing Pr by Ps),

$$\mathrm{Ps}(X = x_{\text{th}})\mathrm{H}(S|X = x_{\text{th}}) \underset{<}{\overset{>}{?}} \mathrm{Ps}(Y = y_{\text{th}})\mathrm{H}(S|Y = y_{\text{th}}), \tag{25}$$

and using the soft probability definition for uniform distribution, we can express Eq. (25) as follows:

$$\frac{1}{2}\mathrm{H}(S|X = x_{\text{th}}) \cdot \bar{0} \underset{<}{\overset{>}{?}} \frac{1}{4}\mathrm{H}(S|Y = y_{\text{th}}) \cdot \bar{0}. \tag{26}$$

Now all we need is to calculate the entropy in the threshold (which is not infinitesimal), and we can conclude which has the better information gain. In the case where $\mathrm{H}(S|X = x_{\text{th}}) = \mathrm{H}(S|Y = y_{\text{th}})$, we can say that X is the feature we should split our data according to as $\frac{1}{2} \cdot \bar{0} > \frac{1}{4} \cdot \bar{0}$, i.e., $\mathrm{IGs}(X) > \mathrm{IGs}(Y)$.

To further understand the applicability of our suggested solutions, we will analyze the next example of numeric of an electric product warranty.

3.2.3 Example: Electric Product Warranty

We would like to predict if an electric product will fail before the end of its 1 year warranty or not. Suppose that we have the following data to analyze:

- Two features that are the "Average Power Consumption" A and the "Peak Power Consumption" B. Due to production process, both have uniform distributions: $A \sim U(0.790, 0.800)$[watt] and $B \sim U(0.900, 1.000)$[watt].
- Denote the random variable F as the failure of an electric product ("$F = 1$" or simplify $F \rightarrow$ failure, "$F = 0$" or simplify $\bar{F} \rightarrow$ no failure), and assume that we have enough samples to calculate the conditional distributions $\Pr(F|A = a)$ and $\Pr(F|B = b)$.

Form the measurement, we have a Heaviside probability function as follows for the average power consumption:

$$\Pr(F|A=a)=\begin{cases}0 & \text{if } a>0.975\\ 1 & \text{if } a\leq 0.975\end{cases} \tag{27}$$

Same goes for the peak power consumption, but the threshold is in different values as follows:

$$\Pr(F|B=b)=\begin{cases}0 & \text{if } b>0.950\\ 1 & \text{if } b\leq 0.950.\end{cases} \tag{28}$$

So clearly when we want to split our data by each parameter we know which value will be our splitting threshold, for the average power consumption, we will choose $A=0.795$, and for the peak power consumption, we will choose $B=0.950$. In this example, for simplicity, we will assume that the initial entropy of F is $\mathrm{H}_0(F)=0.7$. Now we need to calculate which feature gives us the most information gain as the C4.5 algorithm defines. But if we look closely at the threshold numbers, we can see that they are exactly in the middle of the possible values range of each feature, and when we know they are uniformly distributed, we get the following equation:

$$\Pr(A>0.795)=\Pr(A\leq 0.795)=\Pr(B>0.950)=\Pr(B\leq 0.950)=0.5. \tag{29}$$

From that the conditional probability is a Heaviside function, we have the following IG of for each feature:
for the average power consumption

$$\begin{aligned}&\mathrm{IG}(W)\\ &=\mathrm{H}_0(F)-[\Pr(A\leq 0.795)\mathrm{H}(F|A\leq 0.795)+\Pr(A>0.795)\mathrm{H}(F|A<0.795)]\\ &=0.7,\end{aligned} \tag{30}$$

and for the peak power consumption

$$\begin{aligned}&\mathrm{IG}(B)\\ &=\mathrm{H}_0(F)-[\Pr(B\leq 0.950)\mathrm{H}(F|B\leq 0.950)+\Pr(B>0.950)\mathrm{H}(F|B<0.950)]\\ &=0.7.\end{aligned} \tag{31}$$

As we can see, we get equilibrium between the two IGs, and in classic approaches, we can randomly choose the splitting by any feature. On the other hand, using the "Soft Decision Tree" approach, we suggested in this work we can get more information to get better defined decision. To calculate the soft IG, we first need to calculate the soft probability in the equal "=" branch (the soft branch of the tree) as

follows:

$$\begin{aligned} \mathrm{Ps}(A = 0.795) &= \frac{1}{(0.800) - (0.790)} \cdot \bar{0} = 100 \cdot \bar{0} \\ \mathrm{Ps}(B = 0.950) &= \frac{1}{(1.000) - (0.900)} \cdot \bar{0} = 10 \cdot \bar{0}. \end{aligned} \tag{32}$$

For simplicity, we will assume that $\mathrm{H}(F|A = 0.795) = \mathrm{H}(F|B = 0.950) = 0.5$, as we have a theoretical data set. Now we can calculate the soft information gain of both features.

For the average power consumption,

$$\begin{aligned} \mathrm{IGs}(A) &= \mathrm{H}_0(F) - \mathrm{Ps}(A = 0.795)\mathrm{H}(F|A = 0.795) \\ &\quad - [\mathrm{Ps}(A < 0.795)\mathrm{H}(F|A < 0.795) + \mathrm{Ps}(A > 0.795)\mathrm{H}(F|A < 0.795)] \\ &= -50 \cdot \bar{0} \dot{+} 0.7, \end{aligned} \tag{33}$$

and for the peak power consumption,

$$\begin{aligned} \mathrm{IGs}(B) &= \mathrm{H}_0(F) - \mathrm{Ps}(B = 0.950)\mathrm{H}(F|B = 0.950) \\ &\quad - [\mathrm{Ps}(B < 0.950)\mathrm{H}(F|B < 0.950) + \mathrm{Ps}(B > 0.950)\mathrm{H}(F|B < 0.950)] \\ &= -5 \cdot \bar{0} \dot{+} 0.7. \end{aligned} \tag{34}$$

Now when we compare the results, we can clearly see that $\mathrm{IGs}(B) > \mathrm{IGs}(A)$ so we will decide to split the data according to the feature, average power consumption A.

4 Conclusions

In the classical probability, in continuous random variables, there is no distinguishing between the probability involving strict inequality and non-strict inequality. Moreover, a probability involves equality collapse to zero, without distinguishing among the values that we would like that the random variable will have for comparison. Soft numbers assist us to distinguish between the probability involving strict inequality and non-strict inequality, and among the values that we would like that the random variable, by generating soft zeros multiples of the PDF observations.

We introduced an approach to implement C4.5 algorithm as an example for a soft decision tree, but with discrete labels and continuous features. We demonstrated with some numerical examples how the C4.5 algorithm would decide to split the data based on a "soft branch" created by a soft probability of event that would collapse absolutely to zero, and without that branch, we might decide to split the data just arbitrarily.

5 Suggestions for Future Research

We suggest extending the notion of soft probability covered in this work by generalizing to the followings: continuous random vectors, mixed random variable (that has continuous and discrete distribution, i.e., non-piecewise constant CDF but with discontinuity), random vector with discrete, continuous, and mixed random variables, etc. We also suggest exploring the implementation of decision trees with soft numbers into other decision trees algorithms and also to consider each case whenever the labels/features are either discrete variables or continuous variable given being within intervals and single points.

We also suggest exploring the soft logic in general and soft probability in particular in additional topics in information theory, data mining, machine learning, computability, meta-verse technology, cyber-physical system (CPS), etc. We also suggest involving the views of the theory of consciousness in the mentioned above scientific and technological topics, with the concept of the zero axis presents the inner world or virtual world, and the one axis the real world (see paragraph below Eq. (78) for more details). We believe that with soft logic (and soft probability) we can incorporate the spiritual concept of consciousness, which presents inner/virtual world or the zero axis, into the scientific and technological topics in the real world or the one axis.

Acknowledgements This paper was supported by the Koret Foundation grant for Smart Cities and Digital Living 2030 bestowed upon the universities of Stanford and Tel Aviv. We are happy to express our thankfulness and gratitude for this support.

We also want to thank Sagy Naim for his idea to implement soft decision trees based on C4.5 algorithm and for writing the related Sect. 3.2.

Appendix 1: Complement, Union, Intersection, and Conditional Soft Probabilities

Complement, Unions, and Intersections

Recall that a probability of A^c, a complement of the event A, is given by

$$\Pr(A^c) = 1 - \Pr(A). \tag{35}$$

A soft probability of a complement is defined similarly as follows:

$$\mathrm{Ps}(A^c) = 1 - \mathrm{Ps}(A). \tag{36}$$

Therefore, we have the following probability complement for a continuous random variable X:

$$\begin{aligned} \mathrm{Ps}(X \neq x) &= 1 - \mathrm{Ps}(X = x) \\ &= [-f_X(x)]\bar{0}\dot{+}1. \end{aligned} \tag{37}$$

This equation distinguishes among different values of x for the event $X \neq x$ to be with almost surely with probability 1 due to the soft zero term $[-f_X(x)]\bar{0}$. This equation is analogous to the event $X \neq x$ to have zero probability almost surely, correct by the soft zero term $[-f_X(x)]\bar{0}$.

In order to analyze unions and intersections, we need to consider two cases: unions and intersections among singletons events $X = x$, $X = y$, etc., unions and intersections between a singleton event $X = x$ and a range event, e.g., $a \leq X \leq b$.

For all $x \neq y$, we have that the events $X = x$ and $X = y$ are disjoint, and for a union, we have

$$\begin{aligned} \mathrm{Ps}(X = x \cup X = y) &= \mathrm{Ps}(X = x) + \mathrm{Ps}(X = y) \\ &= [f_X(x) + f_X(y)]\bar{0}. \end{aligned} \tag{38}$$

For an intersection, we have

$$\mathrm{Ps}(X = x \cap X = y) = \mathbb{1}_{x=y} f_X(x)\bar{0}, \tag{39}$$

where the indicator $\mathbb{1}_{x=y}$ is zero in the case that $x \neq y$. More generally, we have the following soft probabilities for the following set $\{x_i\}_{i=1}^n$ with distinct values:

$$\mathrm{Ps}\left(\bigcup_{i=1}^{n} X = x_i\right) = \sum_{i=1}^{n} \mathrm{Ps}(X = x_i) = \left[\sum_{i=1}^{n} f_X(x_i)\right]\bar{0}, \tag{40}$$

and

$$\mathrm{Ps}\left(\bigcap_{i=1}^{n} X = x_i\right) = \mathbb{1}_{x_i = x_j}^{\forall i,j \in \{1,2,\ldots,n\}} f_X(x_i)\bar{0}. \tag{41}$$

In order to analyze unions and intersections, between a singleton event $X = x$ and a range event, e.g., $a \leq X \leq b$, we need to distinguish among x's values that are either between a and b or not. Moreover, we need to distinguish between the strict inequality case $a < X < b$ and the non-strict inequality $a \leq X \leq b$. For simplicity, assume $a < b$, and without loss of generality (WLOG), assume $x \neq a$ and $x \neq b$.

For the strict inequality case $a < X < b$, we have the union

$$\mathrm{Ps}(X = x \cup a < X < b) = \mathbb{1}_{x \notin (a,b)} f_X(x)\bar{0}\dot{+}[F_X(b) - F_X(a)], \tag{42}$$

and for the intersection,

$$\mathrm{Ps}(X = x \cap a < X < b) = \mathbb{1}_{x \in (a,b)} f_X(x)\bar{0}. \tag{43}$$

This union is a soft number when x is not in the interval (a, b) and a real number when it does. This intersection is a soft zero when x is in (a, b) and an absolute zero when it does not.

For the non-strict inequality case $a \leq X \leq b$, we have the union

$$\begin{aligned} &\mathrm{Ps}(X = x \cup a \leq X \leq b) = \\ &[\mathbb{1}_{x \notin [a,b]} f_X(x) + f_X(a) + f_X(b)]\bar{0}\dot{+}[F_X(b) - F_X(a)], \end{aligned} \tag{44}$$

and for the intersection

$$\mathrm{Ps}(X = x \cap a \leq X \leq b) = [\mathbb{1}_{x \in [a,b]} f_X(x)]\bar{0}. \tag{45}$$

The two terms $f_X(a) + f_X(b)$ in Eq. (44) are added to the soft zero part, due to Eq. (40).

Recall the relation between a union and an intersection of two events A, B, according to De Morgan's Law; we have

$$\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B). \tag{46}$$

It can be shown that the soft probabilities in Eqs. (42)–(45) hold for De Morgan's Law Eq. (46). For example, $A = \{X = x\}$, $B = \{a \leq X \leq b\}$, and $x \notin [a, b]$, and we have

$$\begin{aligned} &\mathrm{Ps}(X = x \cup a \leq X \leq b) = \\ &\mathrm{Ps}(X = x) + \Pr(a \leq X \leq b) - \Pr(X = x \cap a \leq X \leq b). \end{aligned} \tag{47}$$

The LHS is

$$[f_X(x) + f_X(a) + f_X(b)]\bar{0}\dot{+}[F_X(b) - F_X(a)],$$

and the RHS is

$$f_X(x)\bar{0} + \left[\{f_X(a) + f_X(b)\}\,\bar{0}\dot{+}\{F_X(b) - F_X(a)\}\right] - 0,$$

so that we obtain the LHS to be equal to the RHS, and thus we have a "Soft De Morgan's Law"

$$\mathrm{Ps}(A \cup B) = \mathrm{Ps}(A) + \mathrm{Ps}(B) - \mathrm{Ps}(A \cap B). \tag{48}$$

In the next subsection, we show the results for a conditional of soft probability, referring to Kolmogorov definition and Bayes theorem.

Conditional Probability

Recall Kolmogorov definition for conditional probability

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)}, \tag{49}$$

and for Bayes theorem

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)}. \tag{50}$$

We define a "Soft Conditional Probability" similarly, e.g., for $x, y \in S_X$, let $A = \{X = x\}$, $B = \{X = y\}$, and at the LHS of Kolmogorov definition Eq. (49), we have

$$\frac{\mathrm{Ps}(X = x \cap X = y)}{\mathrm{Ps}(X = y)} = \frac{\mathbb{1}_{x=y} f_X(x)\bar{0}}{f_X(y)\bar{0}} = \frac{\mathbb{1}_{x=y} \cdot \bar{0}}{1 \cdot \bar{0}}. \tag{51}$$

With a definition of $\frac{1\cdot\bar{0}}{1\cdot 0} = 1$ and $\frac{0\cdot\bar{0}}{1\cdot 0} = 0$, the conditional soft probability is given by

$$\mathrm{Ps}(X = x|X = y) = \mathbb{1}_{x=y}. \tag{52}$$

In this case, we have a trivial equality with optional real values 0 or 1. For comparison with Bayes theorem Eq. (50),

$$\frac{\mathrm{Ps}(X = y|X = x)\mathrm{Ps}(X = x)}{\mathrm{Ps}(X = y)} = \frac{\mathbb{1}_{y=x} f_X(x)\bar{0}}{f_X(y)\bar{0}} = \mathbb{1}_{x=y}. \tag{53}$$

Now we consider $x, y \in S_X$; let $A = \{X = x\}$, $B = \{a \leq X \leq b\}$, with $x, a, b \in S_X$ such that $a < b$,$x \neq a$ and $x \neq b$. At the LHS of Kolmogorov definition Eq. (49), we have

$$\frac{\mathrm{Ps}(X = x \cap a \leq X \leq b)}{\mathrm{Ps}(a \leq X \leq b)} = \frac{\mathbb{1}_{x\in[a,b]} f_X(x)\bar{0}}{[f_X(a) + f_X(b)]\bar{0}\dot{+}[F_X(b) - F_X(a)]}. \tag{54}$$

When applying Bayes theorem Eq. (50), we have

$$\frac{\mathrm{Ps}(a \le X \le b|X = x)\mathrm{Ps}(X = x)}{\mathrm{Ps}(a \le X \le b)} = \frac{[\mathrm{Ps}(a \le x \le b|X = x)]f_X(x)\bar{0}}{[f_X(a) + f_X(b)]\bar{0}\dot{+}[F_X(b) - F_X(a)]}, \tag{55}$$

where $\mathrm{Ps}(a \le x \le b|X = x) = \mathrm{Ps}(a \le x \le b) = \mathbb{1}_{x\in[a,b]}$. Both Kolmogorov theorem form and Bayes theorem form are equal, and therefore,

$$\mathrm{Ps}(X = x|a \le X \le b) = \frac{\mathbb{1}_{x\in[a,b]}f_X(x)\bar{0}}{[f_X(a) + f_X(b)]\bar{0}\dot{+}[F_X(b) - F_X(a)]}. \tag{56}$$

We can simplify the RHS by the property

$$\frac{A\bar{0}}{B\dot{+}C\bar{0}} = \frac{A\bar{0}}{B\dot{+}C\bar{0}} \cdot \frac{B\dot{+}(-C)\bar{0}}{B\dot{+}(-C)\bar{0}} = \frac{AB\bar{0}}{B^2} = \frac{A\bar{0}}{B},$$

and we have the following conditional soft probability with a given non-strict inequality condition:

$$\mathrm{Ps}(X = x|a \le X \le b) = \frac{\mathbb{1}_{x\in[a,b]}f_X(x)\bar{0}}{F_X(b) - F_X(a)}, \tag{57}$$

and for a given strict inequality condition, we have

$$\mathrm{Ps}(X = x|a < X < b) = \frac{\mathbb{1}_{x\in(a,b)}f_X(x)\bar{0}}{F_X(b) - F_X(a)}. \tag{58}$$

The meaning of these last two equations is that we have a soft zero when the observation x makes sense (i.e., between a and b), and it is an absolute zero if x makes no sense (i.e., not between a and b), due to the indicator in the numerator. In addition, division by the denominator $F_X(b) - F_X(a) \in (0, 1)$ makes higher probability than the unconditional probability, which make sense since we have an additional information regarding to the random variable X to be between a and b. In the next subsection, we extend the notion of soft probability for 2 continuous random variables, based on a Soft De Morgan's Law equation (48).

Extension of Soft Probability for 2 Dimensions

Suppose that X and Y are two continuous random variables. By the regular De Morgan's Law equation (46), we can decompose the regular probability object

$\Pr(X \leq x, Y \leq y)$ into a sum of the following probabilities:

$$\begin{aligned}&\Pr(X \leq x, Y \leq y) = \\ &\overbrace{[\Pr(X < x, Y = y) + \Pr(X = x, Y < y) + \Pr(X = x, Y = y)]}^{0} \\ &+ \Pr(X < x, Y < y),\end{aligned} \tag{59}$$

such that each of the first three terms in the bracket collapses to zero in the classical probability. We define the soft probability object $\text{Ps}(X \leq x, Y \leq y)$ in 2 random variables based on a Soft De Morgan's Law equation (48) as follows:

$$\begin{aligned}&\text{Ps}(X \leq x, Y \leq y) = \\ &[\text{Ps}(X < x, Y = y) + \text{Ps}(X = x, Y < y) + \text{Ps}(X = x, Y = y)] \\ &+ \text{Ps}(X < x, Y < y).\end{aligned} \tag{60}$$

In this case, we define the first three terms in the bracket as the following soft zero objects in terms of the CDF $F_{X,Y}(x, y)$ and the PDF $f_{X,Y}(x, y)$:

$$\text{Ps}(X < x, Y = y) = \frac{\partial F_{X,Y}(x, y)}{\partial y} \cdot \bar{0}, \tag{61}$$

$$\text{Ps}(X = x, Y < y) = \frac{\partial F_{X,Y}(x, y)}{\partial x} \cdot \bar{0}, \tag{62}$$

$$\text{Ps}(X = x, Y = y) = \frac{\partial F_{X,Y}(x, y)}{\partial x \partial y} \cdot \bar{0} = f_{X,Y}(x, y) \cdot \bar{0}. \tag{63}$$

The last term is a regular probability along the 1-axis, i.e.,

$$\text{Ps}(X < x, Y < y) = \Pr(X < x, Y < y) = F_{X,Y}(x, y), \tag{64}$$

so that $\text{Ps}(X \leq x, Y \leq y)$ equals to the following soft number:

$$\begin{aligned}&\text{Ps}(X \leq x, Y \leq y) = \\ &\left[\frac{\partial F_{X,Y}(x, y)}{\partial x} + \frac{\partial F_{X,Y}(x, y)}{\partial y} + f_{X,Y}(x, y)\right] \cdot \bar{0} \\ &\dot{+} F_{X,Y}(x, y).\end{aligned} \tag{65}$$

Now, we want to construct the soft probability objects $\text{Ps}(X \leq x, Y < y)$ and $\text{Ps}(X \leq x, Y = y)$ (by symmetry, we can construct $\text{Ps}(X < x, Y \leq y)$ and $\text{Ps}(X = x, Y \leq y)$ accordingly). Based on a Soft De Morgan's Law equation (48), we construct the soft probability $\text{Ps}(X \leq x, Y < y)$ similarly as follows:

$$\mathrm{Ps}(X \leq x, Y < y) = \frac{\partial F_{X,Y}(x, y)}{\partial x} \cdot \bar{0} \dot{+} F_{X,Y}(x, y). \tag{66}$$

Therefore, we can distinguish among the soft probabilities: $\mathrm{Ps}(X \leq x, Y \leq y)$, $\mathrm{Ps}(X < x, Y < y)$, $\mathrm{Ps}(X \leq x, Y < y)$ and $\mathrm{Ps}(X < x, Y \leq y)$. Similarly, we have

$$\mathrm{Ps}(X \leq x, Y = y) = \left[\frac{\partial F_{X,Y}(x, y)}{\partial y} + f_{X,Y}(x, y) \right] \cdot \bar{0}, \tag{67}$$

that is a soft zero. In the next section, we define soft expectation, soft variance, and soft entropy.

Appendix 2: Presentation of Soft Numbers

According to traditional mathematics, the expression 0/0 is undefined, although in fact the whole set of real numbers could represent this expression, since $a \cdot 0 = 0$ for all real numbers a. This observation opens a new area for investigation, which is a part of what it is called in [5] a "Soft Logic" that refers to a new axis, "a continuum of multiples of zeros" with distinction between a positive zero "+0" and a negative zero "−0" (see also [6] and [7]).

Soft Number: Definitions and Axioms

A new object $\bar{0}$ is symbolized in order to generate a continuum of multiples of zeros $a\bar{0}$ on a "$\bar{0}$" axis, where a is a real number. An object $a\bar{0}$ denotes "soft zero," while the object $\mathbf{0} = 0 \cdot \bar{0}$ denotes "absolute zero." The object $\bar{1}$ denotes the real axis (i.e., contains multiples of "ones," $b\bar{1}$), and parallel to the "$\bar{0}$" axis. For simplicity, the symbol $\bar{1}$ is omitted during computations. The following axioms and definitions are developed for soft zeros for all real numbers a and b:

Axiom 1 (Distinction) $a \neq b \Rightarrow a\bar{0} \neq b\bar{0}$.

Definition 1 (Order) $a < b \Rightarrow a\bar{0} < b\bar{0}$.

Axiom 2 (Addition) $a\bar{0} + b\bar{0} = (a + b)\bar{0}$.

Axiom 3 (Nullity) $a\bar{0} \cdot b\bar{0} = 0$, *i.e., soft numbers "collapse" to zero under multiplications.*

Axiom 4 (Bridging) *There exists a bridge between a zero axis and a real axis and vice versa, denoted by a pair of a bridge number and its mirror image about the bridge sign. Bridge numbers of a right type*

$$b\bar{1} \perp a\bar{0}$$

and bridge numbers of a left type

$$a\bar{0} \perp b\bar{1}.$$

Axiom 5 (Non-commutativity) *Bridging operator* $\perp$ *does not commute [7], i.e.,*

$$b\bar{1} \perp a\bar{0} \neq a\bar{0} \perp b\bar{1}.$$

Definition 2 (Soft Number) A soft number is defined as a set of the bridge number pairs of opposite types but with the same components—the same zero axis number $a\bar{0}$ and the same real number b:

$$a\bar{0}\dot{+}b = \{a\bar{0} \perp b; b \perp a\bar{0}\}.$$

In the next subsection, we outline some properties of mathematical operations and functions over the soft numbers.

Mathematical Operations and Functions on Soft Numbers

In this section, we outline some mathematical operations over the soft numbers. Suppose $a\bar{0}\dot{+}b, c\bar{0}\dot{+}d \in \mathbf{SN}$ are given soft numbers; then the following mathematical operations hold based on axioms 2 and 3:

- **Addition/subtraction:**

$$(a\bar{0}\dot{+}b) \pm (c\bar{0}\dot{+}d) = (a \pm c)\bar{0}\dot{+}(b \pm d). \tag{68}$$

- **Multiplication:**

$$(a\bar{0}\dot{+}b) \cdot (c\bar{0}\dot{+}d) = (ad + bc)\bar{0}\dot{+}bd. \tag{69}$$

- **Natural power:**

$$(a\bar{0}\dot{+}b)^n = nab^{n-1}\bar{0}\dot{+}b^n. \tag{70}$$

Based on the above equations, every polynomial $P_N(x)$ that operates on every soft number $\alpha\bar{0}\dot{+}x$ is given by

$$P_N(\alpha\bar{0}\dot{+}x) = \alpha P'_N(x)\bar{0}\dot{+}P_N(x), \tag{71}$$

where $P'_N(x)$ denotes the derivative of $P_N(x)$. This notion is generalized for analytic functions $f(x)$ so that

$$f(\alpha\bar{0}\dot{+}x) = \alpha f'(x)\bar{0}\dot{+}f(x). \tag{72}$$

For a continuous random variable X with a CDF F_X and a PDF $f_X = F'_X$, we have the following soft number (cf. Eq. (6)):

$$F_X(\alpha\bar{0}\dot{+}x) = \alpha f_X(x)\bar{0}\dot{+}F_X(x). \tag{73}$$

In the next subsection, we discuss about the soft axis coordinate system.

Soft Axis Coordinate System

We denote the set of all bridge numbers by **BN** and all soft numbers by **SN**. The coordinate system of soft logic is constructed, as presented in Fig. 4. It starts from 0 to 1 horizontally, and then it turns 90° from 1 to infinity.

Remark 1 There exists a one-to-one correspondence between the segment $(0, 1]$ and the segment $[1, \infty)$.

Remark 2 All lines that connect x to $1/x$ (for all non-zero real x) intersect at a single point.

The statements in Remarks 1 and 2 were demonstrated in [5]. This "single point" denotes the beginning of the soft logic coordinate system. We call this point "the absolute zero." The distance from absolute zero to +0 is 1. An extension of this new coordinate system to the negative numbers is implemented in Fig. 5.

In Fig. 5, we have, in addition to the absolute zero **0**, two additional zeros. One zero is opposite to the number -1 and is not identical with the zero opposite to the number $+1$. Hence, we suggest denoting these two different "zeros" as $+\bar{\mathbf{0}}$ and $-\bar{\mathbf{0}}$.

Figure 6 shows the extended coordinate system for positive and negative numbers with an additional line presenting the multiples of zero. The added line is called a zero line or a zero axis, and the multiples on it are called soft zeros or zero axis numbers.

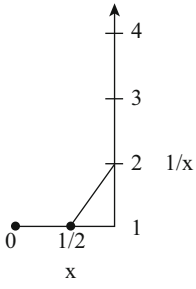


Fig. 4 The Soft coordinate axis

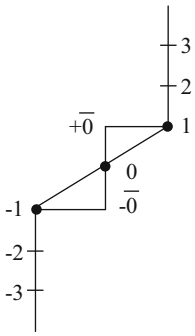


Fig. 5 Distinction between -0 and $+0$

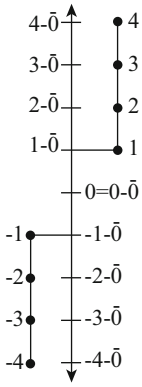


Fig. 6 The extended soft coordinate system

The coordinate system in Fig. 6 allows us to present all the real numbers and all the soft zeros. We now wish to construct a coordinate system for representing various soft numbers, which may be described as an infinite strip as shown in Fig. 4. Because of the soft number duality, we double the strip (Fig. 7). This allows us to represent both elements of a soft number:

$$\begin{aligned} c &= x\bar{0} \perp y, \\ c' &= y \perp x\bar{0}, \end{aligned} \tag{74}$$

where x and y are real numbers. Each of the elements c and c' is a mirror image of the other about the bridge sign. Note that we have expanded the coordinate system in Fig. 6 to the one shown in Fig. 7.

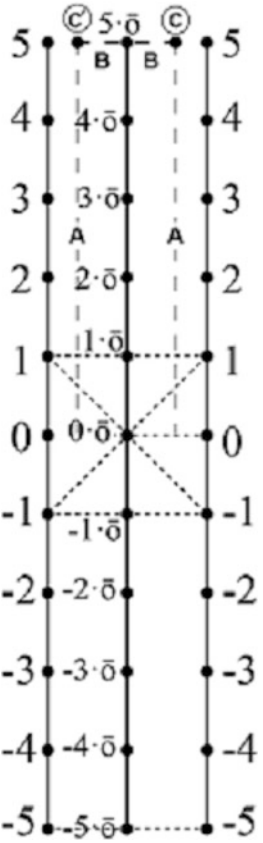


Fig. 7 The complete soft coordinate system

As the infinite strip, presented (partially) in Fig. 7, is intended for the presentation of soft numbers, we call it a *"Soft Numbers Strip"* or briefly, SNS.

Definition 3 (Height and Width of a Point on an SNS) Let C be any point on the SNS:

- The **height of the point** C is the vertical distance from C to the horizontal segment with the absolute zero at its center. This distance is supplied with a plus sign if C is above this segment and with a minus sign if C is below it. The height with a sign is denoted by A.
- The **width of the point** C is the horizontal distance from C to the zero line and is denoted by B.

The definitions above provide every point C on the SNS with two parameters, $A \in \mathbb{R}$ and $B \in [0, 1]$. The condition $A > 0$ is satisfied in the positive part of the SNS, and $A < 0$—in its negative part, or correspondingly, above and below the horizontal segment containing the absolute zero, while on this segment $A = 0$.

For the second parameter B, there is: $B = 0$ on the zero axis, $B = 1$ on the lines bounding the SNS, and otherwise $0 < B < 1$.

If two points c and c' on the SNS are symmetric about the zero axis, they have the same height A and the same width B, i.e., we can symmetrically represent them by the following **BN**s:

$$\begin{aligned} c &= (1 - B)A\bar{0} \perp BA, \\ c' &= BA \perp (1 - B)A\bar{0}. \end{aligned} \tag{75}$$

Therefore, to define a presentation of soft numbers $x\bar{0}\dot{+}y$ by symmetric pairs (**SP**s) of points on the SNS, we have to define a correspondence between these numbers and the pairs of real numbers $(A, B) \in \mathbb{R} \times [0, 1]$ (denoted as **SP**), so that

$$\begin{aligned} x\bar{0}\dot{+}y &= \{c, c'\} \\ &= (1 - B)A\bar{0}\dot{+}BA. \end{aligned} \tag{76}$$

Hence, by a coefficients comparison of the real part and the soft part:

$$\begin{aligned} x &= (1 - B)A \\ y &= BA, \end{aligned} \tag{77}$$

or equivalently, after solving for the **SP**, (A, B)

$$\begin{aligned} A &= x + y \\ B &= \frac{y}{x + y}. \end{aligned} \tag{78}$$

It can be proven that there is an algebraic isomorphism between the bridge numbers $b\bar{0} \perp a$ and dual numbers developed by Clifford [2] with the form $a + b\varepsilon$, where $\varepsilon^2 = 0$ but $\varepsilon \neq 0$. The main difference between ε in dual numbers and $\bar{0}$ is the realization and geometrical interpretation of $\bar{0}$ as an extension of the number 0 on a continuous line. This line can be a model of the inner world, while $\bar{1}$ is a model of the real world. The bridge between them enables us to treat the concept of consciousness with mathematical tools. Another difference is the possibility, in Soft logic, of developing a soft curve [7].

One of our major topics for investigation in further research is the connection of soft numbers to Möbius strip. In order to describe the geometry of Möbius strip with soft numbers, we suggest modifying the soft coordinate system in Fig. 7 by alternating the sign of left vertical line.

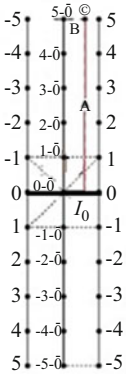


Fig. 8 The alternative soft coordinate system

The horizontal line I_0 in Fig. 8 can represent the connection line where the edges of a straight strip are twisted and attached together to create a Möbius strip. One of the suggestions to define a point on the Möbius strip with soft numbers is that $c = (1 - B)A\bar{0} \perp BA$ is located in the front of this page, while $c' = BA \perp (1 - B)A\bar{0}$ is located behind this page. This setup demonstrates locally existence of two sides of Möbius strip. However, it is known that Möbius strip has globally one side. Moreover, if we start walking vertically from the point c (A units from I_0 and B units from the zero axis) on the front of this page, we will pass through the point behind this page but across the point c' and ($-A$ units from I_0 and B units from the zero axis). When we keep walking on that point, we will go back to the starting point c'. Because of this phenomenon, we are motivated to explore the possibility to represent a soft number with more than two symbols.

During exploration, it is suggested to change Definition 3 width of the point in the SNS will be between -1 and $+1$, i.e., $B \in [-1, 1]$. Given SNS height $A \in \mathbb{R}$ and SNS width $B \in [-1, 1]$, the SNS point $c = c(A, B)$ is defined as follows:

$$\begin{aligned} &\text{For } B \in [0, 1], \\ &\qquad c = (1 - B)A\bar{0} \perp BA, \text{ located in front of page,} \\ &\qquad c' = BA \perp (1 - B)A\bar{0}, \text{ located } \textbf{behind} \text{ this page,} \\ &\qquad x\bar{0}\dot{+}y = \{c, c'\}, \end{aligned} \tag{79}$$

and

$$\begin{aligned} &\text{For } B \in [-1, 0], \\ &\qquad c = (1 + B)A\bar{0} \perp BA, \text{ located in front of page,} \\ &\qquad c' = BA \perp (1 + B)A\bar{0}, \text{ located } \textbf{behind} \text{ this page,} \\ &\qquad x\bar{0}\dot{+}y = \{c, c'\}, \end{aligned} \tag{80}$$

or shortly in terms of absolute value of B:

$$\begin{aligned} c &= (1-|B|)A\bar{0} \perp BA, \text{ located in front of page,} \\ c' &= BA \perp (1-|B|)A\bar{0}, \text{ located } \textbf{behind} \text{ this page,} \\ x\bar{0}\dot{+}y &= \{c, c'\}. \end{aligned} \tag{81}$$

Hence, by a coefficients comparison of the real part and the soft part in Eq. (81):

$$\begin{aligned} x &= (1-|B|)A \\ y &= BA, \end{aligned} \tag{82}$$

or equivalently, after solving for the **SP**, (A, B), using some arithmetic of absolute value and sign functions:

$$\begin{aligned} A &= \text{sign}(x) \cdot (|x| + |y|) \\ B &= \frac{y}{A} = \text{sign}(x) \cdot \frac{y}{|x| + |y|}. \end{aligned} \tag{83}$$

The conjecture is with Eqs. (82)–(83) and Fig. 8, the geometry of Möbius strip with soft numbers can be defined more properly.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Wadsworth Int. Group, 2017. https://doi.org/10.1201/9781315139470.

[2] W. Clifford, "Preliminary Sketch of Biquaternions,". *Proceedings of the London Mathematical Society*, vol. s1–4, no. 1, pp. 381–395, Nov.1871. https://doi.org/10.1112/plms/s1-4.1.381.

[3] O. Fivel, M. Klein, and O. Maimon, "Decision trees with soft numbers," *International Journal of Circuits, Systems and Signal Processing*, vol.15, pp. 1803–1816, 2022. https://doi.org/10.46300/9106.2021.15.194.

[4] A. I. Khuri, "Applications of Dirac's delta function in statistics," *International Journal of Mathematical Education in Science and Technology*, vol. 35, no. 2, pp. 185–195, 2004. https://doi.org/10.1080/00207390310001638313.

[5] M. Klein and O. Maimon, "Axioms of soft logic," *p-Adic Numbers, Ultrametric Analysis and Applications*, vol. 11, no. 3, pp. 205–215, 2019. https://doi.org/10.1134/s27070046619030038.

[6] M. Klein and O. Maimon, "The dynamics in the soft numbers coordinate system," *Journal of Advances in Mathematics*, vol. 18, pp. 1–17, 2020. https://doi.org/10.24297/jam.v18i.8531.

[7] M. Klein and O. Maimon, "Fundamentals of soft logic," *New Mathematics and Natural Computation*, vol. 17, no. 03, pp. 703–737, 2021. https://doi.org/10.1142/s1793005721500356.

[8] J. Li and J. Chen, "Appendix A: Dirac delta function," in *Stochastic Dynamics of Structures*, J. Li and J. Chen, Eds., John Wiley & Sons, Ltd, 2009. pp. 343–348. https://doi.org/10.1002/9780470824269.app1.

[9] J. R. Quinlan, *C4. 5: Programs for Machine Learning*. Los Altos: Morgan Kaufmann, 1993.

[10] J. R. Quinlan, “Induction of decision trees,” *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986. https://doi.org/10.1007/BF00116251.

[11] L. Rokach and O. Maimon, “Decision trees,” in *Data mining and knowledge discovery handbook*, O. Maimon and L. Rokach, Eds., Springer, 2005, p. 165–192. https://doi.org/10.1007/0-387-25465-X_9.

[12] L. Rokach and O. Z. Maimon. *Data mining with decision trees: theory and applications*. World Scientific, 2015, vol. 81. https://doi.org/10.1142/9097.

[13] R. Zanetti and K. Tuggle, “A novel Gaussian mixture approximation for nonlinear estimation,” in *2018 21st International Conference on Information Fusion (FUSION)*, IEEE, 2018, pp. 1100–1106. https://doi.org/10.23919/ICIF.2018.8455485.

Quality Assessment and Evaluation Criteria in Supervised Learning

Amichai Painsky

1 Introduction

During the past decades, there has been a tremendous growth in the theory and practice of supervised learning. A broad variety of views have led to the development of a rich algorithmic toolbox. However, it is not always clear which algorithm is most suitable for a given problem. Therefore, it is essential to assess the performance of an algorithm with respect to a given problem and choose the most appropriate approach. Typically, the first step toward this goal is the choice of a performance measure. This task raises several basic questions, as different criteria focus on different aspects of the algorithm. Over the years, performance measures have received a great amount of attention [14]. As a consequence of the multidisciplinary nature of machine learning tasks, different measures have been influenced by approaches from a variety of disciplines, including statistics, medicine, and signal processing. This chapter reviews prominent performance measures and discusses a variety of issues one encounters during an evaluation study. The focus of our discussion is quality assessment for a given supervised learning algorithm. However, we also discuss important aspects in the design of the algorithm toward the end of this chapter.

When choosing an appropriate evaluation criterion, several questions come to play. The first issue is naturally concerned with the type of learning algorithm to be evaluated. We distinguish between classification and regression problems. In classification, the target variable takes values over a finite set of labels (for example, classification between dogs and cats). In regression, the target variable takes values over a continuous set. For example, students' height. Notice that ordinal

A. Painsky (✉)
Tel Aviv University, Tel Aviv, Israel
e-mail: amichaip@tauex.tau.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_9

variable problems (students' grades from A to F) are mostly considered as variants of regression.

The output of a regression algorithm is an estimate of the target, which typically takes values over the same support. On the other hand, classification algorithms are categorized into hard-decision and soft-decision methods. A hard-decision algorithm outputs a fixed class label from the domain of the target. A soft-decision algorithm provides a score that corresponds to a "level of confidence" in the class label. For example, a classical decision tree would output either a dog or a cat (henceforth, a hard decision), while logistic regression (soft decision) assigns a numerical score (0.2 for dog and 0.8 for cat). Although scores usually hold a probabilistic interpretation, it is not a requirement; scores may hold any numerical value that allows us to rank the classes accordingly. Notice that a hard decision may be obtained by applying a simple threshold to any soft-decision algorithm. For example, we declare a cat label if its score is greater than 0.5. In this sense, soft-decision algorithms are considered more informative. It is important to emphasize that there exist additional types of classifiers that may utilize different performance criteria. For example, *probably approximately correct* (PAC) methods output probabilistic scores on a space of classifiers [35]. In this chapter, we limit our attention to most commonly used classifiers that correspond to either a soft decision or a hard decision.

The following sections discuss different measures for assessing the performance of supervised learning algorithms, their respective strengths and limitations, and their suitability to a particular train or test setting. In Sect. 2, we consider hard-decision classification algorithms, which only utilize information from the confusion matrix. In Sect. 3, we extend our discussion to scoring classifiers and present popular graphical evaluation criteria. In Sect. 4, we study probabilistic classifiers and introduce important connections to basic concepts in decision theory, such as loss functions, divergence measures, and regret. In Sect. 5, we review evaluation criteria for regression problems. We conclude this chapter with a brief discussion in Sect. 6.

1.1 Notations and Definitions

We consider the following standard supervised learning framework. Let $\{x_i, y_i\}_{i=1}^n$ be a given set of samples (train set), where x_i is a vector of features (explanatory variables) and y_i is the corresponding target (independent variable). In supervised learning, we seek a mapping $\hat{y} = f(x)$ such that $\hat{y}$ is "close" to y (denoted as $\hat{y} \approx y$), for a future pair $\{x, y\}$. A classification problem refers to the case where y takes values over a finite set $y \in \mathcal{Y}$ (for example, $y \in \{0, 1\}$ in binary classification). In regression problems, we consider $y \in \mathbb{R}$. For simplicity, we assume that $x \in \mathbb{R}^d$, where d is the dimension of x (the number of explanatory variables). We study different evaluation criteria for the accuracy of the mapping $\hat{y} \approx y$. Notice that in most typical setups, we follow the standard assumption that

both the train set $\{x_i, y_i\}_{i=1}^{n}$ and the future observations $\{x, y\}$ are independent and identically distributed (i.i.d.) drawn from an unknown distribution p_{XY}. However, this assumption is not required in most parts of this chapter. It is important to emphasize that the discussed performance measures are usually evaluated on a holdout set that is independent of the train set (for example, a test set or a validation set, $\{x_i, y_i\}_{i=1}^{m}$), to avoid over-fitting.

2 Hard-Decision Classifiers

Let $\hat{y} = f(x)$ be a supervised learning algorithm for a classification problem where $y \in \mathcal{Y}$. A hard-decision classifier refers to a mapping function that outputs a single element in the domain of y. Formally, $f : \mathbb{R}^d \to \mathcal{Y}$. The most popular performance measure for hard-decision classifiers is the zero-one loss, $l(y, \hat{y}) = \mathbb{1}(\hat{y} \neq y)$. That is, the loss of misclassifying an example (assigning a wrong class label) is one, and zero otherwise. The zero-one loss implicitly assumes that the cost of misclassifying an example is symmetric. For example, the cost of misclassifying a dog as a cat is identical to misclassifying a cat as a dog. However, this is not the case in many real-world problems. For example, consider the problem of tumor classification. Here, the cost of misclassifying a benign tumor is typically greater than misclassifying a malignant tumor. Therefore, we usually distinguish between different types of errors and evaluate learning algorithms accordingly [3, 12]. The *confusion matrix* (formally defined in Sect. 2.1) summarizes all possible classification events. It provides an overview of the performance of a given algorithm, from which different evaluation criteria may be derived. In this sense, the confusion matrix captures all possible events in hard-decision classification.

2.1 The Confusion Matrix

Let $y \in \mathcal{Y} = \{1, \ldots, B\}$ be a multi-class label. Let f be a hard-decision classifier that takes values over the same domain, $f(x) \in \mathcal{Y}$. Denote the confusion matrix (which depends on f) as $C \triangleq C(f)$. The confusion matrix C is a square $B \times B$ matrix for a data set with B classes. Each element c_{kl} of the confusion matrix denotes the number of examples with a label k that the classifier f assigns to the class l. Formally, for a set of m samples $\{x_i, y_i\}_{i=1}^{m}$,

$$c_{kl} = \sum_{i=1}^{m} \mathbb{1}(y_i = k \wedge f(x_i) = l). \tag{1}$$

As mentioned above, the confusion matrix captures all the information induced by a hard-decision classifier, typically evaluated on a holdout data set that is independent of the train set. Specifically, we have that:

- $\sum_l c_{kl}$ is the total number of samples of class k in the data set.
- $\sum_k c_{kl}$ is the total number of samples assigned to the class l by the classifier.
- The diagonal entries c_{kk} correspond to the correctly classified samples. Therefore, $\sum_k c_{kk}$ is the total number of correctly classified samples.
- The non-diagonal entries are the misclassification events. Hence, $\sum_{k \neq l} c_{kl}$ is the total number of misclassified samples.

2.2 The Binary Confusion Matrix

Let us now focus on the important binary classification case, where $B = 2$. The binary (two class) classification problem is perhaps the most common setting in which the performance of a learning algorithm is being measured. For this reason, different notations are borrowed from different scientific fields. In this chapter, we follow the standard computer science literature and denote the two-class labels as *negative* ($y_i = 0$) and *positive* ($y_i = 1$). This notation implies some symmetry between the labels. On the other hand, the statistics literature denotes a negative label as *Null*, while a positive label is the *Alternative*. This notation implies that the two classes are not symmetric, in the sense that positive labels correspond to discoveries, while the negative labels are no-discoveries. The binary confusion matrix is a 2×2 matrix. The entries on the diagonal correspond to correct classification events, where c_{11} is called the *true negative* (TN), while c_{22} is the *true positive* (TP). Similarly, c_{12} is the *false positive* (FP), and c_{21} is the *false negative* (FN). In words, false positive refers to the event of a positive decision that is wrong (the true class is negative). Therefore, the total number of negatives is $\mathrm{N} = \mathrm{TN} + \mathrm{FP}$, while the total number of positives is $\mathrm{P} = \mathrm{TP} + \mathrm{FN}$. Let us illustrate these notations by a simple example. Consider a population of 100 individuals, of which 10 are cancer patients. A classifier f is trained to identify the patients from the entire population. Its performance is summarized in a confusion matrix, as appears in Table 1.

The classifier correctly identified all cancer patients ($\mathrm{FN} = 0$) while mistakenly classified 5 healthy individuals as cancer patients ($\mathrm{FP} = 5$). This example emphasizes the asymmetric nature of many classification problems. The classifier is designed to find all suspected cancer patients, typically for further examination. In this sense, misclassifying a cancer patient as healthy is a "worse" mistake than the

Table 1 Binary classification confusion matrix

Label	Negative	Positive	Total
No cancer	$\mathrm{TN} = 85$	$\mathrm{FP} = 5$	$\mathrm{N} = 90$
Cancer	$\mathrm{FN} = 0$	$\mathrm{TP} = 10$	$\mathrm{P} = 10$

opposite. Therefore, the classifier strives to maximize the number of TPs, even in the expense of some FPs. In the statistics literature, TP is denoted as *true discovery*, to emphasize its role in the problem. The FP is denoted *false discovery* or *type I error*. Similarly, TN is *true null* and FN is *false null* or *type II error*.

2.3 Performance Measures

The confusion matrix captures all classification events for a given learning algorithm. However, we are typically interested in a single scalar measure that summarizes the performance of the algorithm. In this section, we review common performance measures that are drawn from the confusion matrix.

2.3.1 Accuracy and Error Rate

Accuracy is probably the most common performance measure in classification problems. It captures the portion of correctly classified samples, from the total number of samples. Formally, the accuracy is defined as

$$\mathrm{Acc} = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}(y_i = f(x_i)) = \frac{1}{m}\sum_{k=1}^{B} c_{kk}. \tag{2}$$

Notice that the accuracy is proportional to the trace of the confusion matrix. Similarly, the complementary error rate corresponds to the portion of misclassified samples,

$$\mathrm{R} = 1 - \mathrm{Acc} = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}(y_i \neq f(x_i)) = \frac{1}{m}\sum_{k\neq l} c_{kl}. \tag{3}$$

For example, in the cancer patient classification problem (Table 1), we have $\mathrm{Acc} = 0.95$ and $\mathrm{R} = 0.05$. Notice that these measures are not restricted to binary classification, as they hold the same interpretation in the multi-class case.

The accuracy and error rate gain their popularity from their simplicity and intuitive nature. Here, all classification events are considered equally important and are summarized into a single scalar term. In the example above, the accuracy is the sum of TNs and TPs, even though identifying the cancer patients is the major concern of this problem. This emphasizes the major disadvantage of these measures, as they fail to convey varying degrees of importance in different classes. Furthermore, accuracy and error rate are effective measures when the proportion of instances belonging to different classes in the set is relatively balanced (i.e., similar for different classes). As soon as the distribution begins to skew in the direction

of a particular class, this class becomes more dominate and henceforth results in a biased measure. For example, an accuracy of 0.95 can be attained with 10/10 TP and 85/90 TN (as in the example above), or alternatively with 5/10 TP and 90/90 TN. Obviously, the difference in TP is quite dramatic, but the reported accuracy remains. We study skew and cost consideration in a greater depth in Sect. 2.4.

2.3.2 True-Positive Rate, False-Positive Rate, and Likelihood Ratios

The most natural measure for asymmetric problems (where the classes are not considered equally important) is arguably the *true-positive rate*. Although this notion may be a bit misleading in the multi-class setup (indeed, which class is considered "positive" among the many classes?), it is relatively more intuitive in the binary classification setup. The true-positive rate (TPR) refers to the proportion of true positives from all the positive samples. It is formally defined as

$$\text{TPR} = \frac{\sum_{i=1}^{m} \mathbb{1}(y_i = 1 \wedge f(x_i) = 1)}{\sum_{i=1}^{m} \mathbb{1}(y_i = 1)} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{4}$$

Similarly, we define the *false-positive rate* (FPR) as the proportion of false positives from all the negative samples,

$$\text{FPR} = \frac{\sum_{i=1}^{m} \mathbb{1}(y_i = 1 \wedge f(x_i) = 0)}{\sum_{i=1}^{m} \mathbb{1}(y_i = 0)} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{5}$$

True- and false-positive rates generally form a complement pair of reported performance measures. Moreover, we can obtain the same measures on the negative class, in the form of *true-negative rate* (TNR) and *false-negative rate* (FNR), respectively.

$$\text{TNR} = \frac{\text{TN}}{\text{FP} + \text{TN}} = 1 - \text{FPR}, \quad \text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}} = 1 - \text{TPR}. \tag{6}$$

In signal detection theory, the true-positive rate is also known as the *hit rate*, whereas the false-positive rate is referred to as the *false-alarm rate* or *the fall-out*. The true-positive rate of a classifier is also referred to as the *sensitivity* of the classifier. The term has its origin in the medical domain, in which the metric is typically used to study the effectiveness of a clinical test in detecting a disease. The process of evaluating the test in the context of detecting a disease is equivalent to investigating how sensitive the test is to the presence of the disease. That is, how many of the positive instances (actual disease cases) can the test successfully detect? The complement metric to this focuses on the proportion of negative instances (e.g., control cases or healthy subjects) that are detected. This metric is called the *specificity* of the learning algorithm. Hence, sensitivity is generally considered in terms of the positive class, while specificity corresponds to the negative class.

So what is the meaning of these measures? Consider the example in Table 1. The objective of a classifier is to correctly identify all cancer patients, for a follow-up physician examination. The presented classifier attains a sensitivity of 1. This means that all cancer patients were identified as expected. In addition, the specificity of the classifier is $85/90 = 0.94$. It means that only 6% of the healthy patients were mistakenly referred to a physician.

The pair TPR and FPR provides a complete picture of the two classes correctly classified. Unlike accuracy, they do so separately, for each individual class. As a result of the class-wise consideration, the measures reduce the dependency on uneven class distribution. However, while accuracy is reported by a single scalar value, the pair TPR and FPR require two scalars, which makes a biased comparison. Therefore, we introduce a measure that combines TPR and FPR to a single value. The *positive likelihood ratio* and *negative likelihood ratio* are defined as follows:

$$\mathrm{LR}_+ = \frac{\mathrm{TPR}}{\mathrm{FPR}}, \quad \mathrm{LR}_- = \frac{\mathrm{FNR}}{\mathrm{TNR}}. \tag{7}$$

In words, following our example in Table 1, LR_+ summarizes how many times more likely cancer patients are to have a positive prediction than healthy individuals. Equivalently, LR_- summarizes how many times less likely cancer patients are to have a negative prediction than healthy individuals. A higher positive likelihood and a lower negative likelihood mean better performance on positive and negative classes, respectively, so we want to maximize LR_+ and minimize LR_- simultaneously. In practice, likelihood ratios reaching values greater than 10 and lower than 0.1 provide acceptably strong evidence [8]. An additional approach to combine the TNR with FPR (or alternatively $\mathrm{TNR} = 1 - \mathrm{FPR}$) is by taking their geometric mean,

$$G_{mean} = \sqrt{\mathrm{TPR} \times \mathrm{TNR}}. \tag{8}$$

This measure takes into account the relative balance of the classifier's performance on both the positive and negative classes. G_{mean} equals 1 if and only if $\mathrm{TPR} = \mathrm{TNR} = 1$. For all other values, the geometric mean provides a relative balance between the two.

2.3.3 Positive- and Negative-Predictive Values

The true-positive and false-positive rates focus on the proportion of correct classification events from the population of each class. A different perspective on the performance of a learning algorithm is its *predictive value*, the proportion of correct classification events from all classification events of each class. Formally, in the binary classification problem, the positive- and negative-predictive values (PPV and NPV, respectively) are defined as follows:

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{NPV} = \frac{\text{TN}}{\text{FN} + \text{FN}}. \tag{9}$$

PPV measures how many samples are truly positive from all the samples that are assigned to this class. Hence, it measures how "precise" the algorithm is with respect to the class. Therefore, PPV is also referred to as *precision*. The complement of PPV is the *negative-predictive* value (NPV), which measures the proportion of correctly assigned negative examples. In the cancer patients example (Table 1), we have that PPV $= 10/15$ and NPV $= 85/85 = 1$. It means that all the individuals who are classified as healthy are indeed healthy, while only 10 of the 15 individuals who are classified as cancer patients are indeed cancer patients. The interpretation of these results suggests that the classifier is "bad" for identifying cancer patients and "good" for identifying healthy individuals. In fact, this is an alternative view of our objective—we would like to dismiss all the healthy individuals and assign the suspected cancer patients for further examination. Notice that as with previously discussed measures, PPV and NPV are more intuitive in binary classification. These notations may also be extended to multi-class classification, but in this case the notation of "positive" and "negative" need to be clear from the context.

2.3.4 Precision, Recall, and F-measure

Finally, we consider an approach that combines measures from the two previous sections. Here, we focus on the PPV together with the sensitivity (TPR) of the classifier. These are typical statistics of interest in domains such as information retrieval, in which we are interested not only in the proportion of relevant information identified, but also in investigating the actually relevant information from samples tagged as relevant [9]. In the information-retrieval domain, the PPV and TPR are generally referred to as *precision* and *recall*, respectively. In the cancer patients example (Table 1), the precision value is $10/15$ and the recall value 1. It means that all the cancer patients are identified correctly, which are 10 of the 15 positive classifications that are made. This result reflects the strength of the classifier over the positive class. However, it does not provide sufficient information on the negative class. This is quite disturbing since the same values of precision and recall may be obtained no matter what proportion of patients labeled as healthy are actually healthy. Nevertheless, in application where the focus is mostly on the positive class (as in the information-retrieval case), these measures capture the desired merits. In order to combine the two measures into a single value, we consider their weighted harmonic mean, denoted as the *F-measure*,

$$F_\alpha = \frac{(1-\alpha)(\text{Prec} \times \text{Rec})}{\alpha \cdot \text{Prec} + \text{Rec}}, \tag{10}$$

where Prec and Rec are the precision and recall, respectively, while $\alpha \in \mathbb{R}_+$ is a positive constant. For example, for $\alpha = 1$, we obtain the *balanced F-measure*,

while F_2 and $F_{0.5}$ define different trade-offs between the recall and precision. In our example (Table 1), we have $F_{0.5} = 0.75$, $F_1 = 0.8$, and $F_2 = 0.85$. This means that as α increases, we put more emphasize on the recall value (which equals 1) and attain a greater F-measure. This shows that the choice of the relative weight is quite significant in the evaluation of the F-measure. Unfortunately, in most practical cases, appropriate weights are generally unknown, resulting in a significant limitation.

2.3.5 Choosing a Hard-Decision Measure

As mentioned throughout the previous sections, different measures focus on different aspects of the problem. It is important to emphasize that no single metric is capable of encapsulating all the aspects of interest, even with respect to an individual class. Therefore, choosing a performance measure highly depends on the understanding of the problem. In our example, we saw that measures such as accuracy fail to well represent the performance of the classifier, as the problem is much more sensitive to the positive class. In addition, while some measures provide a single scalar value, the others are typically reported in pairs or require some parametric combination. This makes a biased comparison. A single scalar is obviously less informative. On the other hand, increasing the number of metrics being reported makes it increasingly difficult to interpret the results. Moreover, if we are to report multiple measures, then why not simply return the confusion matrix altogether? We discuss this issue in greater depth in Sect. 3.1. Despite these limitations, the performance measures above are highly popular for assessing the quality of hard-decision classifiers as they perform reasonably well when used in the right context.

2.4 Skew and Cost Considerations

The performance measures discussed in the previous section consider the entries of the confusion matrix and provide corresponding measures, which focus on different aspects of the problem. Yet, several issues that are not directly addressed by the confusion matrix may still raise. The first is class imbalance. This corresponds to the case where the classes significantly differ in size. We briefly mentioned this issue earlier in the chapter, in the context of several performance measures. Here, we suggest a general approach that weights the classes according to their relative sizes. Define the class ratio as the proportion of one population size over the other. Specifically, $\text{ratio}_+ = \text{P/N}$ and $\text{ratio}_- = \text{N/P}$. Then, we may weigh the entries according to their respective class ratios. For instance, in the binary case, the TPR is weighted by ratio_+, while TNR is weighed by ratio_-. This type of evaluation, which takes into account the differing class distributions, is referred to as a skew-sensitive assessment. In the multi-class problem, we may similarly define the ratio

of the class k as $\text{ratio}_k = \sum_l c_{kl} / (\sum_{l,l\neq k} c_{lk} + \sum_{l,l\neq k} c_{ll})$. This corresponds to the number of instances in a class k, over the population of the other classes in the set.

A second issue that is not directly addressed by the confusion matrix is asymmetric misclassification costs. In our example above (Table 1), the cost of misclassifying a cancer patient as healthy is considered significantly greater than the opposite. Notice that this asymmetry is not quantified, and so we simply focus on measures that put more emphasis on the positive class. As with the class imbalance problem, we now suggest a more general approach and consider a weight matrix W that defines the misclassification cost. These costs can be either known a priori or come from domain experts. By doing so we attain weighted variants of the different performance measures described in the previous sections. For instance, the error rate, defined as $\sum_{k\neq l} c_{kl}$, has now a weighed variant, $\sum_{k\neq l} w_{kl} c_{kl}$ [21]. Note that cost considerations need not be the same as skew considerations; misclassification cost may or may not overlap with the presence of class imbalance. Although attempts have been made to integrate the two into *cost ratios*, we do not discuss these here. Interested readers are referred to [13, 4] for a detailed discussion.

3 Scoring Classifiers

Hard-decision classifiers are quite intuitive, as they output a single-class label that may, or may not, equal the true label of the sample. In this section, we extend the discussion to scoring classifiers. These classifiers output a real-valued score to each possible label. The scores do not necessarily hold any probabilistic interpretation, although such probabilistic classifiers may be considered as a special case of scoring classifiers. However, we do assume that the scores are monotone, in the sense that a higher score corresponds to a greater confidence in the predicted label. In many applications, the classification scores are compared to a threshold, to obtain class memberships (hard decision) for the samples. For example, all samples with scores above a given threshold are labeled positive, whereas those with scores below it are labeled as negative. In this sense, hard-decision classifiers are also considered as a special case of scoring classifiers. The process of thresholding classification scores is very common in post-processing of many learning algorithm. As we show throughout this section, the threshold value has a significant effect on the performance of the algorithm and its tuning may be considered part of the algorithm's calibration.

Due to their non-discrete nature, soft-decision classifiers are evaluated by different means than hard-decision classifiers. Here, graphical analysis methods that capture the continuous nature of the classifiers are shown to be very effective tools. Among these, the *receiver operating characteristic* (ROC) analysis is perhaps the most popular approach [13, 10]. In the next section, we discuss ROC analysis and its associated performance measures. Then, we extend the discussion to alternative graphical methods that can be applied, depending on the domain of the application.

3.1 ROC and AUC

Graphical performance measures provide visualization of the classifier's performance over its full operational range. This allows us to study a classifier under different skew ratios and class distribution priors. In this sense, graphical methods are significantly more informative than the confusion matrix and its related measures. However, even in this case, it is typically not easy to decide which classifier is more appropriate than the others. Alternatively, if we have information over the full operating range, it is usually easier to discover areas of optimality. That is, it is easier to identify the skew ratios under which one classifier is superior, compared to its alternatives. ROC has its origin in signal detection theory [27]. It was originally introduced as a tool of setting a threshold (operating point) for the receiver to detect the presence of signal. The selection of the best operating point is typically a trade-off between the hit rate (TPR) and the false-alarm rate (FPR). In the context of learning algorithms, ROC graphs have been used in a variety of ways, mostly in binary classification problems. Here, ROCs are utilized to identify optimal behavior regions, perform model selection and most importantly, and evaluate learning algorithms.

The ROC curve is a plot in which the horizontal axis (the x-axis) denotes the false-positive rate (FPR) and the vertical axis (the y-axis) denotes the true-positive rate (TPR) of a given classifier. Notice that TPR is nothing but the sensitivity of the classifier, whereas FPR is simply 1-TNR. Hence, ROC analysis studies the relationship between the sensitivity and the specificity of the binary classifier. Both TPR and FPR take values on the unit interval. Therefore, the ROC space is defined on the unit square, as shown in Fig. 1. As discussed in Sect. 2.3.2, a hard-decision classifier may be characterized by a pair (FPR,TPR). This means that a hard-decision classifier is represented by a single point in the ROC space. The point (0, 0) denotes a trivial classifier that classifies all the samples as negative and, hence, results in both the TPR and the FPR equal zero. On the other end of the square, the point (1, 1) corresponds to the trivial classifier that labels all the instances as positive and henceforth results in TPR = FPR = 1. The diagonal connecting these two points satisfies TPR = FPR. Hence, the classifiers falling along this diagonal are considered random. Specifically, they assign a positive label to the samples with probability p = TPR = FPR. It follows naturally that the classifiers lying above this diagonal perform better than random. The points (1, 0) and (0, 1) give the other two extremes of the ROC space. The point (1, 0) has FPR = 1 and TPR = 0, meaning that this classifier gets all its estimates wrong. On the other hand, the point (0, 1) denotes the ideal classifier, which gets all the positives right and makes no errors on the negatives. As a rule of thumb, for two hard-decision classifiers hd_a and hd_b on ROC space, hd_a represents a better classifier than hd_b if hd_a is "closer" to the point (0, 1) than hd_b. This corresponds to a greater TPR at a lower cost of FPR. The *operating point* on the ROC space corresponds to a specific decision threshold of the classifier that is used to assign discrete labels to the samples. As mentioned above, samples with scores above the threshold are labeled positive, while the ones below

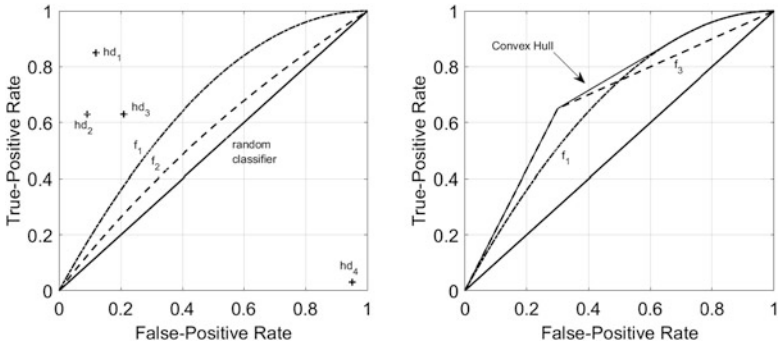


Fig. 1 ROC analysis examples. Left: two soft-decision classifiers f_1 and f_2 and four hard-decision classifiers. Right: example of the ROC convex hull

are labeled negative. The separation between the two classes defines the classifier's hard-decision performance for a particular decision threshold. Hence, each point on the ROC space denotes a specific TPR and FPR of a classifier.

By tuning the decision threshold, one obtains a different pair of TPR and FPR, for each value of the scoring threshold. This results in a continuous curve over the ROC space (for example, see the f_1 and f_2 in Fig. 1). However, this is not necessarily the case in most practical scenarios. Given a finite set of samples m, the number of values on the ROC curves is not greater than m. That is, when the samples are sorted according to their scores, then all the decision thresholds in the interval between two consecutive scores attain the same TPR and FPR. Further, notice that a continuous tuning of the decision threshold is not always possible. For example, decision trees only allow a finite number of thresholds (upper bounded by the number of possible labels over the leaves of the decision tree). Hence, in this setup, the obtained ROC curve consists of step functions.

Let us illustrate the properties of the ROC, for several hard- and soft-decision classifiers. The panel on the left of Fig. 1 summarizes the performance of four hard-decision classifiers (hd_1 to hd_4) and two soft-decision classifiers (f_1 and f_2) on the ROC space. At a first glance, hd_1 seems superior to all other hard-decision classifiers, as it is closer to the optimal point (0, 1). However, notice that if our problem specifies that FPR is the measure of interest, then in some cases hd_2 may be considered more appropriate for the task in hand. Yet, there are several definite conclusions that could be made by observing the ROC. First, hd_3 is inferior to hd_2 as it attains the same TPR at a greater cost (greater FPR). Second, hd_3 is also inferior to hd_1 as hd_1 surpasses it in both merits of interest. Finally, hd_4 seems to be the worst classifier of all, as it is below the random classifier line. However, notice that by flipping its predicted labels (positive becomes negative and negative becomes positive), its performance on the ROC is "mirrored" with respect to the diagonal random classifier line and becomes the best classifier (closest to (0, 1)).

As we compare f_1 to f_2, we see that f_1 is uniformly above f_2, which means that it is superior to it on the entire operational regime. Unfortunately, this is not

always the case. For example, consider the two scoring rules f_1 and f_3 in the right panel of Fig. 1. Here, f_3 dominates f_1 in low FPR values, while f_1 is superior in large FPR. The right panel of Fig. 1 further illustrates an additional ROC concept, the *ROC convex hull* (ROCCH). The ROCCH is a convex hull over the ROC curve. In the case of multiple classifiers, the ROCCH identifies the best classifier(s) for different operating points. The ROC curve of the classifier that lies exactly on the convex hull for a given region of interest denotes the optimal classifier for that operating region. Importantly, the ROCCH can also infer a hybrid classifier that can give optimal performance for different operating points by stochastically weighing different classifiers in the hybrid. Such hybrid classifiers are not in the scope of this chapter. The interested reader is referred to [18] for pointers on the subject.

Although the ROC affords the advantage of visualizing the performance of a classifier over its entire operating range, it does not allow us to quantify this into a single comparable measure. In other words, while a classifier may dominant its competitors over the entire ROC, these cases are somewhat rare. Typically, there is no uniform dominance, and it is difficult to conclude which classifier is superior. Various summary statistics have been proposed to address this shortcoming. The most popular approach is the *area under the curve* (AUC). This measure represents the performance of a classifier averaged over all the possible operational setups. Noting that the ROC space is a unit square, it is clear that AUC $\in [0, 1]$ with the upper bound attained for a perfect classifier. Moreover, notice that a random classifier, represented by the diagonal line across the ROC space, results in AUC $=$ 0.5. Therefore, a classifier with a reasonably better performance than random is expected to attain an AUC value greater than half. It is important to emphasize that different classifiers may attain the same AUC value despite significant differences in their design. For example, a classifier that performs very well on half of the samples of one class, and very poorly on the other half, would also result in AUC $= 0.5$.

AUC was shown to be related to the accuracy (and error rate) of the classifier [22, 5]. Another interpretation of AUC shows its equivalence to the probability that a classifier would rank a randomly chosen positive sample higher than a randomly chosen negative one. In this sense, it is shown to be equivalent to Wilcoxon's Rank Sum test [23] (also known as the Mann–Whitney U-test [17]). Further, the AUC is known to be closely related to the Gini coefficient (a measure of statistical dispersion), satisfying AUC $=$ (Gini $+ 1)/2$ [16].

Different methods have been suggested to evaluate the AUC [16, 10, 17, 2]. Using Wilcoxon's Rank Sum statistic, we may obtain a simple estimate to the AUC as follows. First, we associate a rank in the order of decreasing scores to each of the scores assigned by the classifier. That is, the sample with the highest score is assigned the rank 1. Then, the AUC estimate follows:

$$\text{AUC} = \frac{\sum_{i \in T_p} (R_i - i)}{\text{N} \cdot \text{P}}, \tag{11}$$

where T_p is the set of all positive samples, P and N are the number of positive and negative samples, respectively, and R_i is the rank of the ith sample in T_p.

Similarly to other scalar measures, the AUC loses significant information about the behavior of the learning algorithm over the entire operating range. However, it is argued that the AUC is a preferred measure in setups of imbalanced classes. It is important to mention that AUC has drawn some criticisms over the years [16]. One of the most obvious concerns states that if two ROC curves (of the two different classifiers) intersect, the AUC-based comparison between the classifiers can be relatively uninformative and even misleading [16]. Nonetheless, in the event where the ranking property of the classifier is important (for instance, in information-retrieval systems), AUC is a more reliable measure of classifier performance, as it assesses the ranking capability of the classifier in a direct manner.

Several attempts have been made to extend ROC analysis to multi-class problems [11, 20, 13]. However, this generalization is not quite straightforward. The main reason is that multi-class ROC requires visualization over B-dimensional plots (where B is the number of classes). This is obviously difficult to attain and, more importantly, to interpret. Naturally, the AUC may also be extended to multi-class problems. Here, there exist several generalizations that suffer from varying limitation due to statistical and computation difficulties. The interested reader may refer to [16] and the references therein for further details.

3.2 Additional Graphical Measures

Despite its great popularity, there exist several visualization alternatives to ROC analysis that are mostly application-oriented. One popular example is *lift charts*. Lift charts describe the true positives against the size of the data set required to achieve this number of true positives. That is, the vertical axis of the lift chart is the TP (as opposed to TPR in the ROC), whereas the horizontal axis denotes the number of samples in the data set considered for the specific TP value on the vertical axis. In highly imbalanced data sets, where the number of positive examples is much smaller than the negative samples, the horizontal axes of the lift chart and ROC curves look very similar as do the curves. Lift charts are more common in the business domains. A typical example is direct-mail advertising. Here, very few people respond to this kind of advertising, compared to the number of emails being sent. Therefore, one is interested in the number of emails necessary to attain a given positive effect. *Precision–recall* (PR) curves define another visualization alternative that focuses on the trade-off between the correctly classified positive samples and the number of misclassified negative samples. As their name suggests, PR curves plot the precision of the classifier as a function of its recall. Thus, the curves look different from ROC and lift curves as they have a negative slope. PR curves are mostly common in the information-retrieval field, as discussed in Sect. 2.3.4 in the context of precision, recall, and the F-measure. It is also important to mention that PR curves may sometimes be more appropriate than ROC curves for imbalanced data [7].

Generally speaking, any combination of performance measures that explores the classification trade-off (such as TPR and FPR in ROC analysis) may be represented in graphical manner. Here, we only review the more popular (and insightful) methods. Clearly, there exist additional trade-off curves that may be more suitable for different applications.

4 Probabilistic Classifiers

A scoring classifier assigns a real-valued score to each possible label. As mentioned in Sect. 3, the scores are assumed to be monotone and correspond to the classifier's confidence in the predicted label. In this section, we study a special case of scoring classifiers, in which the scores are probabilistic measures. In words, a probabilistic classifier $f_k(x_i)$ outputs a probability value for each label $y_i = k$, which corresponds to $f_k(x_i) = \mathbb{P}(y_i = k)$. This allows us to study the problem from a probabilistic perspective and introduce important properties for the desired measures.

For the simplicity of the presentation, we focus on single-class classification. Let us first introduce some additional basic notations. We denote a binary probabilistic classifier as $f(x) \triangleq q \in [0, 1]$. In words, q is the probability of the event $y = 1$. The corresponding performance measure is defined as

$$l(y, q) = \mathbb{1}\{y = 0\} l_0(q) + \mathbb{1}\{y = 1\} l_1(q), \tag{12}$$

where $l_k(q)$ is a performance measure associated with the event $y = k$, and $\mathbb{1}\{\cdot\}$ is an indicator function. Several examples of commonly used probabilistic measures are presented in Table 2.

The *quadratic performance measure* (also known as *quadratic loss* or *Brier score*) is perhaps the most intuitive and well-known criterion. It is equivalent to the squared error between the probability estimate q and the sample (which equals zero or one). The *spherical performance measure* is a popular alternative to the quadratic loss. It was first introduced by Roby [29] in the context of psychological

Table 2 Probabilistic performance measures for two-class classification

Performance measure	$l(y, q)$	$r(p, q)$
Quadratic	$y(1-q)^2 + (1-y)q^2$	$(p-q)^2 \quad 7$
Spherical	$\frac{-y(1-q)}{\sqrt{q^2+(1-q)^2}} + \frac{-(1-y)q}{\sqrt{q^2+(1-q)^2}}$	$\sqrt{p^2+(1-p)^2} - \frac{pq+(1-p)(1-q)}{\sqrt{q^2+(1-q)^2}}$
Log-loss	$y \log \frac{1}{q} + (1-y) \log \frac{1}{1-q}$	$p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}$
0-1 loss	$y\mathbb{1}\{q < \text{th}\} + (1-y)\mathbb{1}\{q \geq \text{th}\}$	$(2p-1)\mathbb{1}\{p \geq \text{th}, q < \text{th}\}$ $(2p-1)\mathbb{1}\{p \geq \text{th}, q < \text{th}\}$

testing. It has several interesting geometric properties and a strong connection to the statistical notion of surprise [19]. The *logarithmic loss* (*log-loss*, *logistic loss*, *Bernoulli log-likelihood loss*) is another very popular probabilistic performance measure. The log-loss is a statistical measure that corresponds to the log-likelihood of the estimated parameter q. It is also a staple of information theory [6], as the *self-information* loss function, $-\log p(y)$, defines the ideal code-word length for describing the realization $Y = y$. In this sense, the log-loss corresponds to the amount of information that is necessary to convey the observed samples. The 0-1 loss, which is discussed throughout this chapter, may also be considered as a probabilistic performance measure. Here, the probability value q is compared to a threshold value th, as studied in Sect. 3.

Notice that probabilistic performance measures quantify the discrepancy between the true label y and the corresponding estimate q. This means that a smaller performance measure $l(y, q)$ corresponds to better prediction. This notion is highly related to *scoring rules* that are also used to evaluate probabilistic classifier. However, in scoring rules, a greater score corresponds to better prediction. Therefore, for every performance measure, there exists a corresponding "negative" scoring rule. For example, the logarithmic scoring rule is defined as $y \log q + (1-y)\log(1-q)$, which is exactly the negative log-loss as appears in Table 2.

Probabilistic performance measures have been extensively studied over the years. The statistical framework of this approach allows us to introduce several basic properties. We review and discuss these properties in the following sections.

4.1 Proper Performance Measures

Let us first revisit our model assumptions, as introduced in Sect. 1.1. Assume that the pair x, y is drawn from an unknown distribution P_{XY}. In the binary classification setup, we have that $y \in \{0, 1\}$. Therefore, for a fixed (observed) sample x, the variable Y may be modeled as a Bernoulli distributed variable with an unknown parameter p (where p depends on x). Hence, we may define the corresponding expected performance measure as

$$L(p, q) = E_{Y|X}\left(l(Y, q)|X = x\right) = (1-p)l_0(q) + pl_1(q) \tag{13}$$

for a given q. Notice that $L(p, q)$ only depends on the Bernoulli parameter p and the estimate q. A *proper performance measure* is a performance measure for which the minimizer of $L(p, q)$ is the true underlying model we are to estimate, $p = \arg\min_q L(p, q)$. This property is also known as *Fisher consistency* or *unbiasedness*. A *strictly* proper performance measure means that $q = p$ is a unique minimizer. It is easy to verify that all the performance measures in Table 2 are proper. Further, we may design and characterize additional proper performance measures as shown in [33]. The notion of Fisher consistency is of high interest in

many practical setups. It means that the probabilistic classifier that attains the best performance, on the average, is the true model.

In addition, we define several basic regularity conditions for probabilistic performance measure. We say that a performance measure is *fair* if $l_0(0) = l_1(1) = 0$. This means that there is no loss incurred for perfect prediction. Further, we say that a proper performance measure is *regular* if $\lim_{q \searrow 0} q l_1(q) = \lim_{q \nearrow 1} (1-q) l_0(q) = 0$. Intuitively, this condition ensures that making mistakes on events that never happen should not incur a penalty.

4.2 Regret and Divergence

We now draw our attention to a fundamental connection between probabilistic performance measures and divergence analysis. As mentioned above, a proper performance measure attains its minimal expected value for $q = p$. Denote this minimum as $G(p) \triangleq L(p, p)$. This term is also known as the *generalized entropy function* [15], *Bayes risk* [28] or *Bayesian envelope* [24]. For example, assuming that $l(y, q)$ is the log-loss, then the generalized entropy function is the Shannon entropy, $H(p) = -p \log(p) - (1 - p) \log(1 - p)$. The *regret* is defined as the difference between the expected performance and its minimum. For proper performance measures, we have that

$$r(p, q) = L(p, q) - G(p).$$

Savage [31] showed that a performance measure $l(y, q)$ is proper and regular if and only if $G(p)$ is concave, and for every $p, q \in [0, 1]$, we have that

$$L(p, q) = G(q) + (p - q)G'(q).$$

This property allows us to draw an immediate connection between two-class regret and Bregman divergence. Let $f : \mathbb{S} \to \mathbb{R}$ be a continuously differentiable strictly convex function over some convex set $\mathbb{S} \subset \mathbb{R}^n$. Then, its associated Bregman divergence is defined as $D_f(s||s_0) = f(s) - f(s_0) - \langle s - s_0, \nabla f(s_0) \rangle$ for any $s, s_0 \in \mathbb{S}$, where $\nabla f(s_0)$ is the gradient of f at s_0. By setting $\mathbb{S} = [0, 1]$, we have that $\nabla f = f'$ and $r(p, q) = D_{-G}(p, q)$. This means that the regret of a proper loss function is uniquely associated with a Bregman divergence. An important example is the Kullback–Leibler (KL) divergence, $D_{\mathrm{KL}}(p||q)$, associated with the log-loss.

The correspondence between the regret of proper performance measures and Bregman divergences is arguably one of the more important consequences of the probabilistic evaluation approach. It allows us to adopt well-known results from one field to the other and provide useful insights when choosing an evaluation criterion for the problem in hand. We discuss this in greater detail in Sect. 4.5. Unfortunately, this result only applies to binary classification. There exist several generalizations to

multi-class problems, which mostly focus on *separable Bregman divergences*. See [26] for further details.

4.3 Loss Functions and Performance Measures

Before we proceed, it is important to emphasize that probabilistic performance measures are highly related to *loss functions*. Similarly to probabilistic performance measures, loss functions also measure the discrepancy between the true labels and their corresponding estimates. However, loss functions are typically considered during (or prior to) the design of a classifier. In this sense, choosing a loss function is a prerequisite to the characterization of a learning algorithm, as it explicitly defines its objective. On the other hand, our discussion focuses on the evaluation of a given classifier, after it is fully trained. Technically, there is no major difference between the two. However, there exist several differences in their additional requirements. For example, it is typically desirable for a loss function to have some favorable computational or analytical properties. The reason behind this requirement is quite practical; during the training of the algorithm, the objective needs to be efficiently minimized (that is, within a low computational cost). On the other hand, such computational aspects are not a major concern for performance measures, which are only required to evaluate metrics for a given classifier. Thus, in many practical setups, we may prefer convex and smooth enough (at least twice differentiable) loss functions, for which we can derive simple descent algorithms. This requirement is not of high interest when comparing performance measures. Moreover, notice that while a learning algorithm is designed with respect to a single objective, we may evaluate its performance by multiple measures. This makes the choice of a loss function a major concern. We discuss these issues in Sects. 4.5 and 6.

4.4 Comparing Probabilistic Performance Measures

Proper probabilistic performance measures attain their minimum at $q = p$. Therefore, their corresponding regret equals zero for the optimal classifier. However, how do we compare the results attained by different probabilistic measures, for a non-perfect classifier? First, it is important to note that each of the measures has a different magnitude and location. Further, notice that a proper measure remains proper for any affine transformation. For example, we may scale the quadratic measure in a positive factor and still remain with a proper probabilistic performance measure. Therefore, to compare different measures, it is typically required to normalize them. Several studies have focused on this approach, evaluating the differences and similarities between common measures in different setups and applications. In the following section, we take a broader approach and introduce a universal framework for probabilistic performance measures.

4.5 Universal Performance Measures

To illustrate the notation of universality, let us first introduce a simple example. Consider a weather forecaster that estimates the probability of rain on the following day. Its performance may be evaluated by different performance measures, as discussed throughout this chapter. Choosing a "suitable" measure for this problem is a complicated task that requires domain knowledge or expert's advice. Assuming that the desired measure is known in advance, the weather forecaster may be designed accordingly, to minimize this measure. Unfortunately, we typically do not have this information in advance. Moreover, in many cases, the algorithm is evaluated by multiple criteria. It means that while the forecaster is designed with respect to a single objective, it is expected to perform well with respect to several measures.

A universal performance measure is defined as a measure that dominates a set of alternatives (where the set is as board as possible). In other words, by controlling (minimizing) the universal measure, we implicitly control any measure in its dominated set. Although this notation of universality seems quite restrictive, it can be shown that the log-loss is a universal performance measure with respect to a broad set of probabilistic measures [25]. Specifically, let $l(y, q)$ be a fair, regular, and proper performance measure that is smooth and convex in q (see Sect. 4.4 and [26] for discussions on the smoothness and convexity requirements). Then, for every $p, q \in [0, 1]$, the regret $r(p, q) = D_{-G}(p||q)$ associated with $l(y, q)$ satisfies

$$D_{\mathrm{KL}}(p||q) \geq \frac{1}{C(G)} D_{-G}(p||q), \tag{14}$$

where $D_{\mathrm{KL}}(p||q)$ is the KL divergence and $C(G) > -\frac{1}{2}G''(p)|_{p=\frac{1}{2}}$ is a constant (that does not depend on p or q). Notice that this constant addresses the normalization issue discussed in Sect. 4.4. For example, the bound for the quadratic performance measure shows that $D_{\mathrm{KL}}(p||q) \geq (p-q)^2$. The practical implications of this universality guarantee are quite immediate. Assume that the performance measure according to which a learning algorithm is to be measured with is unknown in advance to the experiment. Then, by minimizing the log-loss, we provide an upper bound on any possible choice of measure, associated with an "analytically convenient" (regular, fair, proper, smooth, and convex) performance measure. This property makes the log-loss a universal choice for classification problems as it governs a large and significant class of measures. It also establishes the underlying connection between the choice of loss function, prior to the design of the learning algorithm, and the applied evaluation criteria, after the algorithm is fully trained. The interested reader is referred to [26] for additional results on the rate of convergence and other properties of the log-loss, compared to alternative measures.

4.6 A Real-World Example

To conclude this section, we illustrate different probabilistic performance measures in a simple real-world example. As discussed in Sect. 4.5, weather forecasters typically assign probabilistic estimates to future meteorological events. The estimates are designed to minimize a performance measure, according to which the weather forecaster is evaluated. However, weather estimates serve a large audience, where different recipients may be interested in different measures. This may cause severe difficulties. For example, by minimizing the quadratic loss, a forecaster may assign zero probability of occurrence to very rare events. This makes sense in the mean square error case but may result in an unbounded logarithmic loss.

In this experiment, we evaluate the performance of a given weather forecaster using the probabilistic performance measures described in Table 2. For the purpose of this experiment, we analyze climatic data that were collected by the Bureau of Meteorology of the Australia's National Meteorological Service.[1] This publicly available data set contains the observed weather and its corresponding forecasts in multiple weather stations in Australia. In our experiment, we focus on the predicted chances of rain (where a rainfall is defined as over 2mm of rain) compared with the true event of rain. Our data set contains 33,134 weather observations and their corresponding forecasts that were collected between the 28th and 30th of April, 2016. Only 9% of the samples correspond to an event of rain. The first row of Table 3 summarizes our results.

The most obvious result of our experiment is that the provided weather forecaster results in an unbounded logarithmic loss. In fact, we observe several instances for which the forecaster predicted zero chance of rain where in fact it eventually rained. Let us revise the suggested forecaster to address this concern. More generally, we would like to design a modified forecaster that is universal with respect to as many as possible evaluation criteria. Notice that the given forecasts are attained by an unknown prediction algorithm that is designed according to a variety of features that are unavailable to us. Therefore, we cannot introduce a novel alternative forecaster but only modify the given forecasts. Thus, we suggest a simple logistic regression, where the target is the observed data and the single feature is the given forecasts. This means that our suggested estimator is $q(x) = \frac{1}{1+e^{-\beta_0-\beta_1(x)}}$, where x is the

Table 3 Weather Forecast Experiment

Weather Forecaster	Quadratic	Spherical	Log-loss	0-1 loss
Australian Forecaster	0.0676	0.0739	∞	0.0898
Modified Forecaster	0.0675	0.0722	0.234	0.0901

[1] http://www.bom.gov.au/climate/data/.

given weather forecast and the β's are the regression parameters. Logistic regression is designed to minimize the log-loss between the observations y and the modified estimates $q(x)$. To avoid over-fitting, we train the regression on a different data set from January 2016. Our results are presented in the second row of Table 3. Notice that our suggested forecaster results in a bounded log-loss, where the other measures remain almost unchanged. This shows that even by applying a simple post-processing routine, we significantly improve the log-loss while controlling a large set of measures (as discussed in Sect. 4.5 and in [36] for the 0-1 loss).

5 Regression Problems

In the previous sections, we discuss a variety of evaluation criteria for classification problems, where the target takes values over a finite set. We now turn to the complementary regression problem. Here, the target takes values over a continuous set, typically the real line $y \in \mathbb{R}$, while the regressor is a mapping from the support of x (for simplicity, we assume $x \in \mathbb{R}^d$) to the support of the target, $f : \mathbb{R}^d \to \mathbb{R}$. As in the classification problems, we denote the estimate of y as $\hat{y} = f(x)$.

The standard approach for evaluating regression models is through additive, residual-based performance measures, such as squared error loss or absolute loss. These measures are attractive from a statistical perspective as they have likelihood interpretations. Further, they often represent the "true" (operational) cost of the prediction error. This property makes them also popular from an engineering (and scientific) view point. In this section, we review some commonly used residual-based performance measures and discuss their properties. The interested reader is referred to [18, 30, 1] for additional residual-based methods and alternative (ranked-based) methods.

5.1 Mean Square Error

The *mean square error* (MSE) is perhaps the most popular evaluation criterion for regression problems. It is formally defined as the squared Euclidean norm between the estimates and the true values, $\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$. The MSE gains its popularity from its favorable analytical and statistical properties; it is a continuous and differentiable metric that only depends on second-order statistics of the variables. However, despite its great popularity, it suffers from several drawbacks. One of the more evident weaknesses of the MSE is its high sensitivity to outliers. Specifically, assume a single bad estimate for an extreme observed value. Here, the squared error will magnify the difference between the two and may skew the metric toward a poor assessment of its performance. That is particularly problematic in the presence of noisy data. It means that even a "good" model may have a high MSE in this situation, so it becomes hard to judge how well it performs. On the other hand,

if all the errors are "small" (for example, smaller than 1), then the opposite effect takes place, and we may underestimate the model's performance. Yet, despite these concerns, the MSEs (and the RMSE, which is simply the squared root of the MSE) are very popular evaluation criteria in many applications.

5.1.1 R-Squared

The *R-squared* (denoted as R^2) is proportional to the ratio between the MSE of a given model and the MSE of a reference model. Typically, the reference model is chosen as the simplest model in hand. For example, the average value of the target $\hat{y}_i = \bar{y} = \sum_{i=1}^{m} y_i$. Formally, $R^2 = 1 - \text{MSE(model)}/\text{MSE(base-model)}$, where $\text{MSE(base-model)} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \bar{y})^2$. R^2 provides a natural referenced framework to compare different models. The greater R^2 is, the better is the model. Obviously, R^2 is closely related to MSE. Its major advantage is its scale-free support, $R^2 \in (-\infty, 1]$, which does not depend on the scale of the problem. It is important to emphasize that our discussion is restricted to performance measures evaluated on an independent holdout set. Therefore, we do explicitly consider issues as the models' degrees of freedom in our measures. Specifically, we do not penalize the model's complexity. Such considerations are addressed, for example, by the adjusted R^2 [34, 32].

5.2 Mean Absolute Value

The *mean absolute error* (MAE) is an average of absolute differences between the target values and the estimates, $\frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|$. It is a linear measure in the sense that all the individual differences are equally weighted. For example, the difference between 10 and 0 is twice the difference between 5 and 0 (as opposed to the MSE). In other words, it penalizes residual errors linearly (and not quadratically). The MAE measures the L_1 distance between the target and the estimate. It is not continuous and henceforth more difficult to minimize during the training of the algorithm. Yet, it plays a major role in many applications as it is less sensitive to outliers.

5.3 Percentage Error

Consider two different setups in which we observe only a single sample. In the first setup, we observe $y = 10$ and estimate $\hat{y} = 9$. In the second, we observe $y = 100$ and estimate $\hat{y} = 99$. In both cases, both the MSE and the MAE equal one. However, it is quite natural to regard a unit error as more significant in the

case where $y = 10$, as opposed to $y = 100$. This leads to the *mean-squared percentage error* (MSPE) and *mean absolute percentage error* (MAPE), defined as $\frac{1}{m}\sum_{i=1}^{m}\frac{(y_i-\hat{y}_i)^2}{y_i}$ and $\frac{1}{m}\sum_{i=1}^{m}\frac{|y_i-\hat{y}_i|}{y_i}$, respectively. Here, the error of each estimate is factored by the true value of the target. This results in a performance measure proportional to the values of the target, and not just the residuals. In this sense, the MSPE and MAPE are considered weighted versions of MSE and MAE, where the weight of each sample is inversely proportional to target.

6 Discussion

In this chapter, we review a variety of performance measures for supervised learning algorithms. For classification problems, we distinguish between hard-decision and soft-decision classifiers. Hard-decision classifiers output a fixed class label from the domain of the target. They are considered less informative than soft classifiers, as they do not provide any notion of confidence to their decision. These classifiers are typically evaluated by statistics derived from the confusion matrix. Soft-decision classifiers provide a score that corresponds to a "level of confidence" in the class label. They are typically evaluated by graphical means (such as ROC curves) that summarize their performance in operational regimes and their corresponding statistics (AUC). In cases where the scores hold a probabilistic interpretation (for example, the "level of confidence" is the estimated probability of the label), then we may evaluate the classifier in a statistical framework. This allows us to introduce fundamental notions such are consistency, rate of convergence, and others. Finally, we discuss popular residual-based performance measures for regression problems and review their properties in terms of computational complexity and robustness.

The main take-home message from this review states that an evaluation measure (or multiple measures) should be selected according to the specific problem in hand. In other words, when assessing the quality of a learning algorithm, one should first take into consideration the objective and the domain knowledge of the problem. For example, a classifier that is designed to identity rare positives (as with the cancer patients example in Sect. 2) shall not be evaluated by the overall error rate. In other cases, where no domain knowledge is available, then one shall consider performance measures that are more robust (or universal) with respect to others. For example, we show that the regret associated with the log-loss controls (bounds from above) a large set of practical probabilistic performance measures.

Finally, it is important to emphasize that the study of performance measures is highly related to loss function analysis. In this sense, the choice of a loss function (a priori to the design of the algorithm) is highly influenced by the evaluation criterion according to which the algorithm is to be measured. This makes the evaluation considerations of high relevance, already during the design of the learning algorithm.

References

1. Jinbo Bi and Kristin P Bennett, *Regression error characteristic curves*, Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 43–50.
2. Andrew P Bradley, *The use of the area under the ROC curve in the evaluation of machine learning algorithms*, Pattern Recognition **30** (1997), no. 7, 1145–1159.
3. Rich Caruana and Alexandru Niculescu-Mizil, *Data mining in metric space: an empirical analysis of supervised learning performance criteria*, Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2004, pp. 69–78.
4. Michelle Ciraco, Michael Rogalewski, and Gary Weiss, *Improving classifier utility by altering the misclassification cost ratio*, Proceedings of the 1st International Workshop on Utility-Based Data Mining, ACM, 2005, pp. 46–52.
5. Corinna Cortes and Mehryar Mohri, *AUC optimization vs. error rate minimization*, Advances in Neural Information Processing Systems, 2004, pp. 313–320.
6. Thomas M Cover and Joy A Thomas, *Elements of information theory*, John Wiley & Sons, 2012.
7. Jesse Davis and Mark Goadrich, *The relationship between precision-recall and ROC curves*, Proceedings of the 23rd International Conference on Machine Learning, ACM, 2006, pp. 233–240.
8. Jonathan J Deeks and Douglas G Altman, *Diagnostic tests 4: likelihood ratios*, BMJ **329** (2004), no. 7458, 168–169.
9. Gianluca Demartini and Stefano Mizzaro, *A classification of IR effectiveness metrics*, European Conference on Information Retrieval, Springer, 2006, pp. 488–491.
10. Tom Fawcett, *ROC graphs: Notes and practical considerations for researchers*, Machine Learning **31** (2004), no. 1, 1–38.
11. ——, *An introduction to ROC analysis*, Pattern Recognition Letters **27** (2006), no. 8, 861–874.
12. César Ferri, José Hernández-Orallo, and R Modroiu, *An experimental comparison of performance measures for classification*, Pattern Recognition Letters **30** (2009), no. 1, 27–38.
13. Peter A Flach, *The geometry of ROC space: understanding machine learning metrics through ROC isometrics*, Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 194–201.
14. Jerome Friedman, Trevor Hastie, and Robert Tibshirani, *The elements of statistical learning*, vol. 1, Springer Series in Statistics New York, 2001.
15. Tilmann Gneiting and Adrian E Raftery, *Strictly proper scoring rules, prediction, and estimation*, Journal of American Statistical Association **102** (2007), no. 477, 359–378.
16. David J Hand and Robert J Till, *A simple generalisation of the area under the ROC curve for multiple class classification problems*, Machine Learning **45** (2001), no. 2, 171–186.
17. James A Hanley and Barbara J McNeil, *The meaning and use of the area under a receiver operating characteristic (ROC) curve.*, Radiology **143** (1982), no. 1, 29–36.
18. Nathalie Japkowicz and Mohak Shah, *Evaluating learning algorithms: a classification perspective*, Cambridge University Press, 2011.
19. Victor Richmond Jose, *A characterization for the spherical scoring rule*, Theory and Decision **66** (2009), no. 3, 263–281.
20. Nicolas Lachiche and Peter A Flach, *Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves*, Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 416–423.
21. Nada Lavrač, Peter Flach, and Blaz Zupan, *Rule evaluation measures: A unifying view*, International Conference on Inductive Logic Programming, Springer, 1999, pp. 174–185.
22. Charles X Ling, Jin Huang, Harry Zhang, et al., *AUC: a statistically consistent and more discriminating measure than accuracy*, IJCAI, vol. 3, 2003, pp. 519–524.
23. Simon J Mason and Nicholas E Graham, *Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation*,

Quarterly Journal of the Royal Meteorological Society: A Journal of the Atmospheric Sciences, Applied Meteorology and Physical Oceanography **128** (2002), no. 584, 2145–2166.
24. Neri Merhav and Meir Feder, *Universal schemes for sequential decision from individual data sequences*, IEEE Transactions on Information Theory **39** (1993), no. 4, 1280–1292.
25. Amichai Painsky and Gregory Wornell, *On the universality of the logistic loss function*, 2018 IEEE International Symposium on Information Theory (ISIT), IEEE, 2018, pp. 936–940.
26. Amichai Painsky and Gregory W Wornell, *Bregman divergence bounds and universality properties of the logarithmic loss*, IEEE Transactions on Information Theory **66** (2019), no. 3, 1658–1673.
27. H Vincent Poor, *An introduction to signal detection and estimation*, Springer Science & Business Media, 2013.
28. Mark D Reid and Robert C Williamson, *Composite binary losses*, Journal of Machine Learning Research **11** (2010), no. Sep, 2387–2422.
29. Thornton B Roby, *Belief states and the uses of evidence*, Behavioral Science **10** (1965), no. 3, 255–270.
30. Saharon Rosset, Claudia Perlich, and Bianca Zadrozny, *Ranking-based evaluation of regression models*, Fifth IEEE International Conference on Data Mining (ICDM'05), IEEE, 2005, pp. 8–pp.
31. Leonard J Savage, *Elicitation of personal probabilities and expectations*, Journal of American Statistical Association **66** (1971), no. 336, 783–801.
32. George AF Seber and Alan J Lee, *Linear regression analysis*, vol. 329, John Wiley & Sons, 2012.
33. Emir H Shuford, Arthur Albert, and H Edward Massengill, *Admissible probability measurement procedures*, Psychometrika **31** (1966), no. 2, 125–145.
34. Henri Theil, *Economic forecasts and policy*, North-Holland Pub. Co., 1961.
35. Leslie G Valiant, *A theory of the learnable*, Communications of the ACM **27** (1984), no. 11, 1134–1142.
36. Tong Zhang, *Statistical behavior and consistency of classification methods based on convex risk minimization*, The Annals of Statistics (2004), 56–85.

Trajectory Clustering Analysis

Yulong Wang and Yuan Yan Tang

1 Introduction

The past years have seen the explosion of trajectory data with the rapid development of surveillance and tracking devices. Trajectory data clustering [1] as an unsupervised learning task has attracted much attention in machine learning, computer vision, and pattern recognition. It also has numerous applications in motion segmentation [2, 3, 4, 5, 6], abnormal detection [7, 8, 9], and traffic monitoring [10, 11, 12].

In recent years, a number of subspace-based trajectory data clustering (STDC) methods have been proposed, including iterative [13, 14, 15], algebraic [16, 17, 18], statistical [19, 20], and spectral-clustering-based methods [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]. These methods first learn an affinity matrix indicating the membership among the data samples and then apply a spectral clustering algorithm [32] to the affinity matrix, obtaining the clustering results. They are mainly different in the process of solving the affinity matrix. For example, the sparse subspace clustering (SSC) method [3] introduces the *sparse representation* (SR) technique and computes the affinity matrix by solving a sparse program problem. The low-rank representation (LRR) [4] and low-rank subspace clustering (LRSC) [23] methods seek the lowest-rank representation in a given or learned dictionary. Starting with the pioneer work of SSC [3] and LRR [4], many variants of SSC and LRR have also been developed. For example, Patel et al. [33] devised a latent space SSC method to simultaneously perform dimensionality reduction and subspace clustering. Feng et

Y. Wang
Huazhong Agricultural University, Wuhan, China
e-mail: ylwang@mail.hzau.edu.cn

Y. Y. Tang (✉)
University of Macau, Avenida da Universidade, Taipa, China
e-mail: yytang@umac.mo

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_10

al. [29] enhanced SSC and LRR by enforcing the block-diagonal structure constraint on the affinity matrix. Lu et al. [34] proposed the correlation adaptive subspace clustering method by using trace lasso to achieve both the sparsity and the grouping effect. Here lasso [35] is the ℓ_1 norm regularized least squares method. Recently, a variety of robust subspace clustering methods are also developed in the presence of random noise or outliers [36, 37]. In addition, local information around each data sample is also used to build the affinity matrix by some recent methods, such as local subspace affinity (LSA) [38], spectral local best-fit flats (SLBF) [39], and locally linear manifold clustering (LLMC) [40]. Most of the codes of the methods above can be found in the homepages of the corresponding authors.

In fact, many popular STDC methods have much in common. First, most of them consist of two main steps, i.e., learning an affinity matrix and then performing spectral clustering. Second, most of them involve solving a category of linear inverse problems. The optimal solutions of these problems are assumed to have low-dimensional structures of the high-dimensional ambient space [41], such as sparsity, low rankness, etc. The commonalities of these methods motivate us to propose a general framework, called ARSC, to provide a unified view of them. Many state-of-the-art STDC (e.g., SSC and LRR) can be regarded as specializations of ARSC with specific choices of the atomic set, loss function, and constraint. The potential benefits of the unification are as follows. First, with the proposed ARSC framework, we can recover many popular state-of-the-art subspace clustering methods, such as SSC and LRR. Second, ARSC can also potentially induce other possible low-dimensional structures and convex norms unexploited so far [41].

Due to the analytical tractability and the low complexity of the resulting algorithms, most SC approaches utilize MSE as the loss function. Nevertheless, since MSE relies on the Gaussianity assumption [42], MSE-based SC methods are sensitive to non-Gaussian noise. To alleviate such limitation, the work [31] proposed to learn the affinity matrix for trajectory clustering by specifying the *minimum error entropy* (MEE) as the loss function and the sparsity-inducing atomic set. One appealing advantage of MEE over MSE is that MEE does not rely on the Gaussianity assumption as MSE and can well deal with both Gaussian and non-Gaussian noise. The experimental results show that MEESSC, termed MEESSC, outperforms state-of-the-art-related SC methods in motion segmentation and face clustering.

2 Preliminaries

2.1 Summary of Main Notations

In this chapter, we use boldface capital letters, boldface lowercase letters, and normal font to represent matrices, vectors, and scalars, respectively. The calligraphic letters are utilized to denote sets. In particular, $\mathbf{I}$ represents the identity matrix, while $\mathcal{I} = \{1, 2, \cdots, N\}$ denotes the set of N integers. The entries of matrices are

represented by utilizing $[\cdot]$ with subscripts. Concretely, for any matrix $\mathbf{M}$, we use $[\mathbf{M}]_{ij}$, $[\mathbf{M}]_{i,:}$, and $[\mathbf{M}]_{:,j}$ to denote as its (i, j)-th entry, i-th row, and j-th column, respectively. With an abuse of notation, for a matrix $\mathbf{M}$, diag($\mathbf{M}$) represents a vector composed of the diagonal entries of $\mathbf{M}$, while for a vector $\mathbf{v}$, diag($\mathbf{v}$) means a square diagonal matrix with the elements of $\mathbf{v}$ on the main diagonal. span($\mathbf{M}$) stands for the linear subspace spanned by the columns of $\mathbf{M}$. We use $\mathbf{X} \in \text{span}(\mathbf{M})$ to indicate that each column of $\mathbf{X}$ belongs to span($\mathbf{M}$). tr($\mathbf{M}$) is the trace of $\mathbf{M}$. The Euclidean inner product between two matrices is written as $\langle \mathbf{M}_1, \mathbf{M}_2 \rangle = \text{tr}\left(\mathbf{M}_1^T \mathbf{M}_2\right)$, where $\mathbf{M}_1^T$ is the transpose of $\mathbf{M}_1$.

Several matrix norms will be utilized. Specifically, the ℓ_1, $\ell_{2,1}$ norms, Frobenius norm, and nuclear norm of a matrix $\mathbf{M}$ are, respectively, defined as $\|\mathbf{M}\|_1 = \sum_{i,j} |[\mathbf{M}]_{ij}|$, $\|\mathbf{M}\|_{2,1} = \sum_j \|[\mathbf{M}]_{:,j}\|_2$, $\|\mathbf{M}\|_F = \sqrt{\sum_{i,j} [\mathbf{M}]_{ij}^2}$, and $\|\mathbf{M}\|_* = \sum_i \sigma_i$, where σ_is are the singular values of the matrix $\mathbf{M}$.

2.2 Problem Statement

Let $\{\mathcal{S}_k\}_{k=1}^K$ be K linear subspaces of $\mathbb{R}^d$. Denote d_k as the dimension of the subspace $\mathcal{S}_k$. Assume that $\mathbf{X}_0$ is a matrix composed of N data samples drawn from a union of the K subspaces. Let $\mathcal{I}_k$ and N_k, respectively, denote the index set and the number of samples belonging to $\mathcal{S}_k$ and $\mathcal{I}_k^c = \mathcal{I} - \mathcal{I}_k$. Then $\{\mathcal{I}_k\}_{k=1}^K$ make up a partition of the set $\mathcal{I} = \{1, 2, \cdots, N\}$, i.e., $\mathcal{I} = \bigcup_{k=1}^K \mathcal{I}_k$ and $\bigcap_{k=1}^K \mathcal{I}_k = \emptyset$. The observation data matrix $\mathbf{X}$ can be expressed as

$$\mathbf{X} = \mathbf{X}_0 + \mathbf{E}_0, \tag{1}$$

where $\mathbf{E}_0$ stands for the noise term. It is worth pointing out that we do not make any predefined assumption of the distribution of the noise $\mathbf{E}_0$. Given only the possibly noisy data matrix $\mathbf{X}$, the goal of subspace clustering is to accurately segment the data from $\mathbf{X}$ into their underlying subspaces, or equivalently, find the accurate partition $\{\mathcal{I}_k\}_{k=1}^K$ of $\mathcal{I} = \{1, 2, \cdots, N\}$.

3 Trajectory Clustering via Atomic Representation

In this section, we present the unifying ARSC framework along with the algorithm and its popular special cases for subspace clustering.

3.1 Atomic Representation

First we introduce the definition of the atomic norm and some assumptions about the atomic representation. Let $\mathcal{A}$ be an origin-symmetric (i.e., $\mathbf{a} \in \mathcal{A}$ *if and only if* (iff) $-\mathbf{a} \in \mathcal{A}$) and possibly infinite set. The definition of atomic norm is introduced as follows.

Definition 1 ([43]) Given the atomic set $\mathcal{A}$, the atomic norm of $\mathbf{x}$ with respect to $\mathcal{A}$ is defined as

$$\|\mathbf{x}\|_{\mathcal{A}} := \inf_{t>0} \{t : \mathbf{x} \in t \cdot \text{conv}(\mathcal{A})\},$$

where $\text{conv}(\mathcal{A})$ is the convex hull of the set $\mathcal{A}$.

The dual norm of $\|\cdot\|_{\mathcal{A}}$ is defined by $\|\mathbf{z}\|_{\mathcal{A}}^* = \sup_{\mathbf{a}\in\mathcal{A}} \langle \mathbf{z}, \mathbf{a} \rangle$, which satisfies $\langle \mathbf{z}, \mathbf{x} \rangle \leq \|\mathbf{z}\|_{\mathcal{A}}^* \|\mathbf{x}\|_{\mathcal{A}}$ according to the definitions of inner product and dual norm [44].

Now we introduce some common examples of the atomic norm, which have attracted massive attention recently in machine learning and signal processing [45, 46, 47]:

- **Sparsity-inducing norm:** The sparsity-inducing atomic set can be written as

$$\mathcal{A}_S = \left\{ \pm \mathbf{E}_{ij} \in \mathbb{R}^{m\times n}, i = 1, 2, \cdots, m, j = 1, 2, \cdots, n \right\},$$

  where $\mathbf{E}_{ij}$ represents the matrix, of which the (i, j)-th entry is 1 and the others are 0s. For any matrix $\mathbf{Z} \in \mathbb{R}^{m\times n}$, we have $\|\mathbf{Z}\|_{\mathcal{A}_S} = \|\mathbf{Z}\|_1$.
- **Low-rankness-inducing norm:** The low-rankness-inducing atomic set can be expressed as

$$\mathcal{A}_L = \left\{ \mathbf{A} \in \mathbb{R}^{m\times n} | \ \text{rank}(\mathbf{A}) = 1, \|\mathbf{A}\|_F = 1 \right\}.$$

  For any matrix $\mathbf{Z} \in \mathbb{R}^{m\times n}$, there holds $\|\mathbf{Z}\|_{\mathcal{A}_L} = \|\mathbf{Z}\|_*$.
- **Group-sparsity-inducing norm:** Consider a partition $\{\mathcal{G}_k\}_{k=1}^K$ of $\mathcal{G} = \{1, 2, \cdots, m\}$, i.e., $\bigcup_{k=1}^K \mathcal{G}_k = \mathcal{G}$ and $\bigcap_{k=1}^K \mathcal{G}_k = \emptyset$. Denote $\mathcal{A}_{\mathcal{G}_k} = \left\{ \mathbf{A} \in \mathbb{R}^{m\times n} | \ \|[\mathbf{A}]_{\mathcal{G}_k,:}\|_F = 1, [\mathbf{A}]_{\mathcal{G}_k^c,:} = \mathbf{0} \right\}$. Here $[\mathbf{A}]_{\mathcal{G}_k,:}$ represents the matrix composed of the rows of $\mathbf{A}$ corresponding to the indexes in $\mathcal{G}_k$. Then the group-sparsity-inducing atomic set [49] can be expressed as $\mathcal{A}_G = \bigcup_{k=1}^K \mathcal{A}_{\mathcal{G}_k}$. For any matrix $\mathbf{Z} \in \mathbb{R}^{m\times n}$, we have $\|\mathbf{Z}\|_{\mathcal{A}_G} = \sum_{k=1}^K \|[\mathbf{Z}]_{\mathcal{G}_k,:}\|_F$, which encourages group sparsity as a penalty function. Note that when the partition of $\mathcal{G}$ has only one element $\mathcal{G}$, the atomic norm $\|\mathbf{Z}\|_{\mathcal{A}_G}$ equals to the Frobenius norm, which is defined as $\|\mathbf{Z}\|_F = \sqrt{\sum_{i=1}^m \sum_{i=1}^n Z_{ij}^2}$, where Z_{ij} denotes the entry of $\mathbf{Z}$ in the i-th row and the j-th column. Thus Frobenius norm is an extreme case of the atomic norm $\|\cdot\|_{\mathcal{A}_G}$.

With the atomic set defined, the *atomic representation* (AR) model can be formulated as follows:

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_{\mathcal{A}}, \quad \text{s.t. } \mathbf{X}_0 = \mathbf{D}\mathbf{Z}, \tag{2}$$

where $\mathbf{D}$ and $\mathbf{Z}$ represent the dictionary and the coefficient matrix, respectively. Since the atomic norm is a general form of ℓ_1 norm and the nuclear norm, the model in Eq. (2) generalizes sparse representation and LRR.

3.2 Algorithm of ARSC

In this section, we utilize the atomic representation to build the general ARSC framework along with the algorithm for subspace clustering. Many state-of-the-art spectral-clustering-based SC algorithms (e.g., SSC and LRR) can be regarded as specializations of ARSC with specific choices of the atomic set, loss function, and constraint.

Algorithm 1 Algorithm of ARSC

Input: The atomic set $\mathcal{A}$, data matrix $\mathbf{X}$, number K of subspaces and parameter λ.
Output: Segmentation of the data.

1: Obtain the representation matrix $\mathbf{Z}^*$ by solving the AR optimization problem in Eq. (4) for clean data or (5) for noisy data.
2: Construct the similarity matrix $\mathbf{W} = |\mathbf{Z}^*| + |(\mathbf{Z}^*)^T|$
3: Apply a spectral clustering algorithm to the similarity matrix $\mathbf{W}$ and segment the data into K clusters.

The first step is to learn an affinity matrix from the observation data using the atomic representation. When the given data are clean, i.e., $\mathbf{X} = \mathbf{X}_0$, the data samples are perfectly lying in the union of the K subspaces. In light of *self-expressiveness property* of the data [21], each data sample can be linearly represented by other samples in $\mathbf{X}$, namely,

$$[\mathbf{X}]_{:,i} = \mathbf{X}[\mathbf{Z}]_{:,i}, \ [\mathbf{Z}]_{ii} = 0, \ i = 1, 2, \cdots, N,$$

or equivalently,

$$\mathbf{X} = \mathbf{X}\mathbf{Z}, \ \text{diag}(\mathbf{Z}) = \mathbf{0}. \tag{3}$$

However, Eq. (3) may have infinitely many solutions when the number of points in a subspace exceeds the dimension of the subspace. Ideally, we expect that the solution $\mathbf{Z}$ of Eq. (3) satisfies the following property: $[\mathbf{Z}]_{ij} \neq 0$ only if the points $[\mathbf{X}]_{:,i}$ and

$[\mathbf{X}]_{:,j}$ are in the same subspace. Such a solution is referred as a *subspace sparse representation* in [21]. To this end, SSC and LRR assume that the ideal solution is the sparsest and has the lowest rank, respectively. In this chapter, we adopt a more general assumption that the ideal solution has the minimal atomic norm with respect to (w.r.t.) a certain atomic set $\mathcal{A}$ among all solutions of Eq. (3). Then the optimization program can be written as

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_{\mathcal{A}}, \quad \text{s.t. } \mathbf{X} = \mathbf{XZ}, \ \text{diag}(\mathbf{Z}) = \mathbf{0}. \tag{4}$$

In practice, the observation data may be corrupted by noise during data collection. Combining Eq. (1) and the assumption that $\mathbf{X}_0$ satisfies the self-expressiveness property, we have

$$\mathbf{X} = \mathbf{XZ} + \mathbf{E},$$

where $\mathbf{E} = \mathbf{E}_0 - \mathbf{E}_0\mathbf{Z}$. For noisy data, we consider the following stable ARSC model to approximately solve the subspace sparse representation:

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_{\mathcal{A}} \ \text{ s.t. } \ \mathcal{L}(\mathbf{X} - \mathbf{XZ}) \leq \epsilon, \ \text{diag}(\mathbf{Z}) = \mathbf{0}, \tag{5}$$

where $\mathcal{L}(\cdot)$ represents the loss function and ϵ is the error tolerance parameter. Using the method of Lagrangian multiplier, we can rewrite Eq. (5) in a regularization form

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_{\mathcal{A}} + \lambda\mathcal{L}(\mathbf{X} - \mathbf{XZ}) \ \text{ s.t. } \ \text{diag}(\mathbf{Z}) = \mathbf{0}, \tag{6}$$

where λ denotes the regularization parameter. Thus the problems in Eqs. (5) and (6) are equivalent. After solving the affinity matrix $\mathbf{Z}$ with Eqs. (4) or (5), we utilize the similarity matrix $\mathbf{W} = |\mathbf{Z}^*| + |(\mathbf{Z}^*)^T|$ to perform spectral clustering [32, 53] and segment the observation data. The entire algorithm of ARSC is summarized in Algorithm 1, which has been implemented in [48].

3.3 Related Works

We now present a discussion about the relationship between the ARSC framework and related spectral-clustering-based methods. We show why these methods can be viewed as special cases of the proposed framework.

SSC [3, 21] This method seeks the affinity matrix by solving the following sparse optimization problem:

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_1 + \lambda\|\mathbf{X} - \mathbf{XZ}\|_F^2, \quad \text{s.t. } \text{diag}(\mathbf{Z}) = \mathbf{0}. \tag{7}$$

Table 1 Some examples of the ARSC framework

Methods	LSR [50]	SSC0 [3]	SSC1 [21]	LRR [4]	MSR [51]	LRSSC [52]
Loss function	MSE	MSE	MSE+ℓ_1	$\ell_{2,1}$	$\ell_{2,1}$	MSE
Atomic sets	$\mathcal{A}_G$	$\mathcal{A}_S$	$\mathcal{A}_S$	$\mathcal{A}_L$	$\mathcal{A}_S, \mathcal{A}_L$	$\mathcal{A}_S, \mathcal{A}_L$

We denote the model (7) as SSC0. It can be seen that SSC0 belongs to the ARSC framework by choosing $\mathcal{L}(\mathbf{E}) = \|\mathbf{E}\|_F^2$ and $\mathcal{A} = \mathcal{A}_S$. By explicitly considering the sparse outlying entries and random noise, the model (7) can be modified as

$$\begin{aligned} &\min_{\mathbf{Z}} \|\mathbf{Z}\|_1 + \lambda_1 \|\mathbf{E}_1\|_1 + \lambda_2 \|\mathbf{E}_2\|_F^2, \\ &\text{s.t. } \mathbf{X} = \mathbf{X}\mathbf{Z} + \mathbf{E}_1 + \mathbf{E}_2, \ \operatorname{diag}(\mathbf{Z}) = \mathbf{0}. \end{aligned} \tag{8}$$

The model (8) is denoted by SSC1 for simplicity. In essence, SSC1 divides the noise term $\mathbf{E} = \mathbf{X} - \mathbf{X}\mathbf{Z}$ into two terms $\mathbf{E}_1$ and $\mathbf{E}_2$ and enforces different loss functions, respectively.

LRR [4, 22] As another example of ARSC, LRR utilizes the $\ell_{2,1}$ norm as loss function and the atomic set $\mathcal{A}_L$

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_* + \lambda \|\mathbf{X} - \mathbf{X}\mathbf{Z}\|_{2,1}. \tag{9}$$

For LRR, the constraint $\operatorname{diag}(\mathbf{Z}) = \mathbf{0}$ is optional, because the minimization of the nuclear norm tends to avoid the trivial solution $\mathbf{I}$ [4]. To remove the possible outlier samples in the given data, LRR also applies a heuristic post-processing step [22]. To distinguish one from another, we denote the LRR method without and with the post-processing step by LRR0 and LRR1, respectively.

LSR [50] and MSR [51] The LSR method adopts the loss function $\mathcal{L}(\mathbf{E}) = \|\mathbf{E}\|_F^2$ and the atomic set $\mathcal{A}_G$ with $\mathcal{I}$ itself as the partition. The MSR method uses the $\ell_{2,1}$ norm as loss function and a combination of $\mathcal{A}_S$ and $\mathcal{A}_L$. Table 1 shows some previous works, including SSC, LRR, and LSR, as special cases of the proposed framework.

4 Minimum Error Entropy-Based Trajectory Clustering Analysis

This section is structured as follows. We first introduce the concept and background of minimum error entropy. Then we introduce a MEE-based sparse subspace clustering (MEESSC) method along with the optimization algorithm [31].

4.1 Minimum Error Entropy

Suppose that E is a scalar random variable with *probability density function* (pdf) $p_E(e)$. Shannon's entropy [54] of E is defined by

$$H_S(E) = \mathbb{E}[-\log p_E(e)] = -\int p_E(e) \log p_E(e) de, \tag{10}$$

while Renyi's entropy [55] of order-α ($\alpha > 0$ but $\alpha \neq 1$) is defined as

$$H_\alpha(E) = \frac{1}{1-\alpha} \log \mathbb{E}[p_E^{\alpha-1}(e)] = \frac{1}{1-\alpha} \log \left(\int (p_E(e))^\alpha \, de \right), \tag{11}$$

where $\mathbb{E}$ denotes the mathematical expectation. Since in reality the pdf of E is often unknown, the entropy of E cannot be directly computed. Nonetheless, if some samples $\{e_i\}_{i=1}^d$ of E are available, we can utilize the Parzen window method [56] to estimate the pdf of E as follows:

$$\hat{p}_E(e) = \frac{1}{d} \sum_{i=1}^{d} g_\sigma(e - e_i), \tag{12}$$

where $g_\sigma(x) := \exp\left(-\frac{x^2}{2\sigma^2}\right)$ represents the Gaussian kernel function and σ is the kernel size. Denote the vector $\mathbf{e} = [e_1, e_2, \cdots, e_d] \in \mathbb{R}^d$. Substituting $\hat{p}_E(e)$ into Eq. (11) and setting $\alpha = 2$ give the following nonparametric estimator of Renyi's entropy of order 2

$$\hat{H}_2(E) = -\log \int \left(\frac{1}{d} \sum_{i=1}^{d} g_\sigma(e - e_i) \right)^2 de = -\log V(\mathbf{e}). \tag{13}$$

Here $V(\mathbf{e})$ is called the *information potential* (IP) [57], which can be explicitly formulated as

$$V(\mathbf{e}) = \frac{1}{d^2} \sum_{i=1}^{d} \sum_{j=1}^{d} g_{\sqrt{2}\sigma}(e_i - e_j). \tag{14}$$

The last equation in (13) uses the fact that the integral of a product of two same Gaussian kernels is another Gaussian kernel with twice the variance (i.e., σ^2) [57]. For ease of presentation, we use σ to replace $\sqrt{2}\sigma$ *hereinafter*. Denote $\phi(\mathbf{e}) := \sum_{i,j} \left(1 - g_\sigma(e_i - e_j)\right)$. Then the minimization of $\hat{H}_2(E)$ is equivalent to the minimization of $\phi(\mathbf{e})$.

The MEE criterion aims at finding a decent estimation of the unknown original signal by minimizing the entropy of the error random variable $E = X - Y$, where

X and Y are the observation signal and the estimation, respectively. An appealing advantage of MEE over MSE is that the optimality criterion is agnostic to the error distribution. Thus, MEE can well deal with both Gaussian and non-Gaussian noise, while MSE relies on the Gaussianity assumption of the error distribution and is sensitive to non-Gaussian noise. In addition, since MSE simply constrains the square difference between X and Y, it only considers the second moment of the error pdf. In contrast, when the error entropy is minimized, all moments of the error pdf are constrained [57]. Therefore, MEE extends MSE as an optimality criterion. This gives rise to the success of MEE in signal processing [57] and feature selection [58].

4.2 MEESSC

By following the ARSC framework, we specify the loss function $\mathcal{L}(\mathbf{X}) = \Phi(\mathbf{X}) =: \sum_{i=1}^{N} \phi\left([\mathbf{X}]_{:,i}\right)$ and the atomic set $\mathcal{A} = \mathcal{A}_S$. Then we have the following optimization program:

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_1 + \lambda\Phi(\mathbf{X} - \mathbf{X}\mathbf{Z}), \quad \text{s.t. } \operatorname{diag}(\mathbf{Z}) = \mathbf{0}. \tag{15}$$

The problem can be solved by optimizing over each column $[\mathbf{Z}]_{:,i}$ of the matrix $\mathbf{Z}$ individually

$$\min_{[\mathbf{Z}]_{:,i}} \|[\mathbf{Z}]_{:,i}\|_1 + \lambda\phi\left([\mathbf{X}]_{:,i} - \mathbf{X}[\mathbf{Z}]_{:,i}\right), \quad \text{s.t. } [\mathbf{Z}]_{ii} = 0. \tag{16}$$

This can be tackled by first solving the following problem:

$$\min_{\mathbf{z}\in\mathbb{R}^{N-1}} \|\mathbf{z}\|_1 + \lambda\phi\left([\mathbf{X}]_{:,i} - \mathbf{X}^{\setminus:,i}\mathbf{z}\right), \tag{17}$$

where $\mathbf{X}^{\setminus:,i}$ represents the submatrix of $\mathbf{X}$ without the i-th column. Then we set $[\mathbf{Z}]_{:,i} = [z_1, \cdots, z_{i-1}, 0, z_{i+1}, \cdots, z_{N-1}]^T$, where $\mathbf{z} = [z_1, z_2, \cdots, z_{N-1}]$. Thus the problem in Eq. (15) can be transformed into N subproblems in the same form

$$\min_{\mathbf{z}\in\mathbb{R}^{N-1}} \phi(\mathbf{x} - \mathbf{D}\mathbf{z}) + \gamma\|\mathbf{z}\|_1, \tag{18}$$

where $\gamma = 1/\lambda$ and $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$. Denote $\tilde{\mathbf{x}} = [\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_{d^2}]$ and $\tilde{\mathbf{D}}$ such that $\tilde{x}_h = x_i - x_j$ and $[\tilde{\mathbf{D}}]_{h,:} = [\mathbf{D}]_{i,:} - [\mathbf{D}]_{j,:}$, where $h = (i-1)d + j$ for $i, j = 1, 2, \cdots, d$. Then we can rewrite $\phi(\mathbf{x} - \mathbf{D}\mathbf{z}) = \sum_{h=1}^{d^2}\left(1 - g_\sigma\left(\tilde{x}_h - [\tilde{\mathbf{D}}]_{h,:}\mathbf{z}\right)\right)$. In light of the half-quadratic method [60], there exists some function ψ such that the problem in Eq. (18) is equivalent to the following problem:

$$\min_{\mathbf{z},\mathbf{p}} J(\mathbf{z},\mathbf{p}) = \sum_h \left(p_h \left(\tilde{x}_h - [\tilde{\mathbf{D}}]_{h,:}\mathbf{z} \right)^2 + \psi(p_h) \right) + \gamma \|\mathbf{z}\|_1, \tag{19}$$

where $\mathbf{p} = [p_1, p_2, \cdots, p_{d^2}]$ is a vector of auxiliary variables. A local minimizer of the problem (19) can be obtained by alternatively updating $\mathbf{p}$, $\mathbf{z}$, and σ. Specifically, at the t-th iteration for fixed $\mathbf{p}$, the problem (19) reduces to

$$\mathbf{z}^{(t+1)} = \arg\min_{\mathbf{z}\in\mathbb{R}^{N-1}} \left\| \sqrt{\mathrm{diag}(\mathbf{p}^{(t)})}\tilde{\mathbf{x}} - \sqrt{\mathrm{diag}(\mathbf{p}^{(t)})}\tilde{\mathbf{D}}\mathbf{z} \right\|_2^2 + \gamma \|\mathbf{z}\|_1,$$

which can be effectively tackled using the *feature-sign search* (FSS) algorithm [59]. Denote the error vector $\mathbf{e} = \tilde{\mathbf{x}} - \tilde{\mathbf{D}}\mathbf{z}^{(t+1)}$ with e_i as the i-th entry of $\mathbf{e}$. Then we calculate the kernel size

$$\sigma = \left(\left\| \tilde{\mathbf{x}} - \tilde{\mathbf{D}}\mathbf{z}^{(t+1)} \right\|_2^2 / 2d^2 \right)^{\frac{1}{2}}$$

and update the auxiliary vector $\mathbf{p}$ by

$$p_h^{(t+1)} = \frac{1}{\sigma^2} g_\sigma(e_i), \ h = 1, 2, \cdots, d^2,$$

where $p_h^{(t+1)}$ stands for the h-th entry of $\mathbf{p}^{(t+1)}$. Algorithm 2 summarizes the complete procedure to solve the problem (19), which has been implemented in [48]. According to the half-quadratic theory [60], the sequence $\left\{ J\left(\mathbf{z}^{(t)}, \mathbf{p}^{(t)}\right), t = 1, 2, \cdots \right\}$ generated by Algorithm 2 converges.

Note that the MEESSC method and the SSC method mainly differ in the loss functions and the subsequent sparse optimization models to solve the affinity matrix. The sparse optimization problems of SSC and MEESSC can be written in a unified form as follows:

$$\min_{\mathbf{z}} \mathcal{L}(\mathbf{x} - \mathbf{D}\mathbf{z}) + \gamma \|\mathbf{z}\|_1, \tag{20}$$

where $\mathcal{L}(\cdot)$ is a general differentiable loss function. Then we have the following proposition to give the optimality conditions of the problem (20). $\mathbf{z}^*$ is the solution of the problem (20) if and only if (i) $\sup_{1\le i\le n} \left| \langle \theta(\mathbf{z}^*), [\mathbf{D}]_{:,i} \rangle \right| \le \gamma$, (ii) $\langle \theta(\mathbf{z}^*), \mathbf{D}\mathbf{z}^* \rangle = \gamma \|\mathbf{z}^*\|_1$, where $\theta(\mathbf{z}^*) = -\frac{\partial \mathcal{L}(\mathbf{x}-\mathbf{y})}{\partial \mathbf{y}}|_{\mathbf{y}=\mathbf{D}\mathbf{z}^*}$. Since the loss function $\mathcal{L}(\cdot)$ is general, Proposition 4.2 generalizes the existing results about the optimality conditions of the MSE-based ℓ_1 minimization problem (also known as lasso) in the literature [61].

Algorithm 2 Solving (19) via half-quadratic method

Input: $\tilde{\mathbf{x}}$, $\tilde{\mathbf{D}}$, and parameter γ.
Output: $\mathbf{z} \in \mathbb{R}^{N-1}$.
Initialization: $\mathbf{p}^{(0)} = [1, 1, \cdots, 1]^T \in \mathbb{R}^{d^2}$ and $t = 0$.
Repeat until convergence:

1: $\mathbf{y}^{(t)} = \sqrt{\text{diag}(\mathbf{p}^{(t)})}\tilde{\mathbf{x}}$ and $\mathbf{A}^{(t)} = \sqrt{\text{diag}(\mathbf{p}^{(t)})}\tilde{\mathbf{D}}$.
2: Solving the following ℓ_1 minimization problem using the *feature-sign search* (FSS) algorithm [59]

$$\mathbf{z}^{(t+1)} = \arg\min_{\mathbf{z}\in\mathbb{R}^{N-1}} \left\|\mathbf{y}^{(t)} - \mathbf{A}^{(t)}\mathbf{z}\right\|_2^2 + \gamma\|\mathbf{z}\|_1.$$

3: Compute $\mathbf{e} = \tilde{\mathbf{x}} - \tilde{\mathbf{D}}\mathbf{z}^{(t+1)}$.
4: Calculate the kernel size $\sigma = \left(\|\mathbf{e}\|_2^2/2d^2\right)^{\frac{1}{2}}$.
5: Update the auxiliary vector $\mathbf{p}$

$$p_h^{(t+1)} = \frac{1}{\sigma^2} g_\sigma(e_i),\ h = 1, 2, \cdots, d^2.$$

5 Experiments

This section aims at evaluating the performance of the MEESSC method for two real-world subspace clustering problems, i.e., motion segmentation and face clustering.

5.1 Experimental Settings

To begin with, we introduce the implementation details of the experiments, including the used datasets, and competing methods and their parameter settings.

For the motion segmentation problem, we utilize the popular Hopkins 155 database [63]. For the face clustering problem, we consider the Extended Yale B dataset [64] and the CMU PIE dataset [65]. The description of these datasets is as follows:

1. Hopkins 155 Database [63]: This database consists of 156 video sequences along with the features extracted and tracked in all the frames. Each video sequence has almost 30 frames and 2 or 3 motions (corresponding to 2 or 3 low-dimensional subspaces). This dataset is publicly available on the website: http://www.vision.jhu.edu/data/hopkins155/.
2. Extended Yale B Database [64]: It contains 38 subjects and 64 frontal-face images per subject under varying illumination. Each cropped facial image has

192 × 168 pixels. This dataset is publicly available on the website: http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html.

3. CMU PIE Database [65]: This database contains 68 subjects. For each subject, facial images are taken with varying poses, illumination, and expressions. We utilize a subset containing the near frontal facial images of pose C05. In this subset, there are 49 images for each subject. The images are resized to have 32 × 32 pixels. This dataset is publicly available on the website: http://www.cs.cmu.edu/afs/cs/project/PIE/MultiPie/Multi-Pie/Home.html.

For comparison purposes, we consider several state-of-the-art subspace clustering algorithms, including SIM (shape interaction matrix) [2], LSA [38], SSC0 [3], SSC1 [21], LRR0 [4], LRR1 [22], and LSR [50]. For each algorithm, we use the codes provided by the corresponding authors. The parameter settings of these algorithms are specified as below. For SIM, we use $r = 4K$ as the estimate of the rank of $\mathbf{X}_0$ for motion segmentation, while $r = 9K$ for face clustering. Here K is the number of subspaces. For LSA, we use $n = 8$ nearest neighbors and $d = 4$ dimension for motion segmentation, while $n = 7$ and $d = 5$ for face clustering. For SSC0 and SSC1, we use the code implemented in an alternating direction method of multipliers (ADMM) framework for efficiency. Specifically, for SSC0, we set the regularization parameter $\lambda = 800/\mu_z$ for motion segmentation and $\lambda = 20/\mu_z$ for face clustering, where $\mu_z := \min_i \max_{j \neq i} |\mathbf{x}_i^T \mathbf{x}_i|$ [21]. Analogously, for SSC1, we choose $\lambda_1 = 0$ and $\lambda_2 = 800/\mu_z$ with the affine constraint $\mathbf{1}^T \mathbf{Z} = \mathbf{1}$ ($\mathbf{1} \in \mathbb{R}^N$ denotes the vector whose entries are 1s.) for motion segmentation, while $\lambda_1 = 20/\mu_e$ and $\lambda_2 = 0$ for face clustering as [21]. Here $\mu_e := \min_i \max_{j \neq i} \|\mathbf{x}_j\|_1$. Thus, for the motion segmentation problem, the difference between SSC0 and SSC1 is that SSC1 uses the affine constraint, while SSC0 does not. For the face clustering problem, SSC0 uses $\|\cdot\|_F^2$ as the loss function and SSC1 uses $\|\cdot\|_1$. For LRR0 and LRR1, we set $\lambda = 4$ for motion segmentation and $\lambda = 0.18$ for face clustering according to [22]. For LSR, we choose $\lambda = 4.6 \times 10^{-3}$ for motion segmentation and $\lambda = 0.4$ for face clustering [50]. For MEESSC, we use the fixed parameter $\gamma = 0.001$ for all experiments.

5.2 Motion Segmentation

In this section, we evaluate the performance of MEESSC and competing methods on the motion segmentation problem. Given the feature points on multiple rigidly moving objects tracked in multiple frames of a video, the motion segmentation aims to separate the feature points according to their underlying objects. We use the Hopkins 155 dataset for this test. Figure 1 shows several sample images from some sequences with tracked feature points superimposed in this database. The feature points from one subject are marked with the same color. Each video sequence in this dataset is a sole data matrix, and thus there are 156 subspace clustering tasks

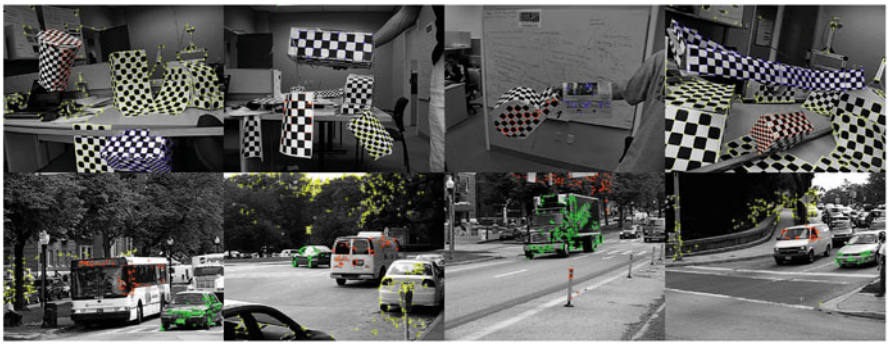


Fig. 1 Sample images from some sequences in the Hopkins 155 dataset with tracked feature points superimposed

Table 2 Clustering error (%) of different algorithms on the 156 sequences of the Hopkins 155 database with the $2F$-dimensional data samples and $4K$-dimensional samples obtained by PCA (principal components analysis). Best results are marked bold

Data dimension	Methods	SIM	LSA	SSC0	SSC1	LRR0	LRR1	LSR	MEESSC
$2F$	Mean	7.10	4.58	7.09	2.23	5.53	2.78	4.01	**1.73**
	Median	1.37	0.57	0.21	**0.00**	0.78	0.00	0.53	**0.00**
	Std	11.83	9.99	12.76	7.26	9.86	6.87	8.30	**6.48**
$4K$	Mean	7.10	4.58	7.17	2.47	5.98	3.65	4.35	**1.90**
	Median	1.37	0.57	0.21	**0.00**	0.78	0.21	0.36	**0.00**
	Std	11.85	9.99	12.88	7.50	10.78	9.16	8.56	**6.71**

in total. The dimension of each data sample is $2F$, where F denotes the number of frames in the current sequence.

We first apply different subspace clustering methods to the original $2F$-dimensional data. To make the results more convincing, we also project the data points of each video sequence into $4K$ dimension using PCA and perform subspace clustering with various methods. Here, K is the number of motions in each sequence. We use $4K$ dimension because the rank of each subspace is at most 4 [22]. The clustering results are shown in Table 2. From the results, we give the following conclusions. First, as a whole, the clustering error of each method increases when the data dimension decreases due to the loss of energy. Second, in both cases, MEESSC outperforms competing methods, having small mean clustering error and standard deviation.

5.3 Face Clustering

In this section, we validate the performance of MEESSC and competing methods for the face clustering problem. Given multiple facial images of multiple subjects, the

goal of face clustering is to separate the facial images according to their underlying subjects. The Extended Yale B database and the CMU PIE database are used to conduct the experiments. First, we apply distinct subspace clustering algorithms on the original face database. Then, to test the robustness of the algorithms, we increase the clustering difficulty by adding random noise to the facial images. Specifically, we consider random contiguous occlusion and random missing entries.

5.3.1 Face Clustering Using Extended Yale B Database

In this subsection, we aim to verify the performance of the clustering algorithms above on the original Extended Yale B database without corruption and pre-processing. For efficiency, we downsample the images to 32×32 pixels. We treat each column-stacked vectorized facial image as a data point with 1024 dimensions. In order to study the effect of number of subjects in the final clustering performance, we apply the clustering algorithms on the first 2, 4, 6, 8, and 10 subjects of the dataset.

Table 3 reports the clustering error of distinct clustering methods. Denote by $\hat{\mathbf{Z}}$ and $\hat{\mathbf{z}}_i$ the representation matrix computed by MEESSC and the i-th column of $\hat{\mathbf{Z}}$. We treat $\mathbf{X}\hat{\mathbf{z}}_i$ as the recovered result of the i-th data sample. Figure 2 shows two representative recovered images by MEESSC in the Extended Yale B database. For the recovery experiment, we use the facial images with 96×84 pixels for decent visual quality. From the results, we have the following conclusions:

- First, in general, when the number of subjects increases, each included clustering method has larger clustering error. This comes from the fact that it is more difficult to learn more subspaces simultaneously from noisy data.
- Second, the clustering error of MEESSC is low under a varying number of subjects. This suggests that MEESSC can well treat the corrupted data in face clustering. The reason may be attributed to that MEESSC can adaptively select the facial images lying in the same subspace of the original corrupted image to well reconstruct it, as shown in Fig. 2.
- Third, SSC1 can greatly enhance SSC0 when the number of subjects exceeds 2. This implies that ℓ_1 norm is a better choice than $\|\cdot\|_F^2$ as a loss function for SSC to cluster noisy facial images in the Extended Yale B database.

Table 3 Clustering error (%) of different algorithms on the Extended Yale B dataset. Best results are marked bold

Methods	SIM	LSA	SSC0	SSC1	LRR0	LRR1	LSR	MEESSC
2 Subjects	7.03	17.19	0.78	0.78	2.34	2.34	3.91	**0.00**
4 Subjects	10.55	56.64	17.19	1.17	16.41	7.81	4.69	**0.39**
6 Subjects	9.90	54.17	16.41	4.95	32.29	5.47	8.07	**1.56**
8 Subjects	28.32	59.18	47.66	7.62	30.86	12.11	12.89	**4.49**
10 Subjects	35.94	69.22	47.19	9.22	37.03	24.84	27.34	**8.44**

Fig. 2 Recovered results of two facial images by MEESSC in the Extended Yale B database. In each subfigure, images from left to right are the original image, the recovered image by MEESSC, and the difference between the original and recovered images

Fig. 3 Top row: some sample facial images with 25% occlusion by an unrelated monkey image from the Extended Yale B database; bottom row: some sample facial images with 20% missing entries in each image from the CMU PIE data set

5.3.2 Face Clustering with Contiguous Occlusion

In this subsection, we evaluate the clustering performance of MEESSC on the Extended Yale B database in the presence of contiguous occlusion. Concretely, for each test image, we simulate the contiguous occlusion by replacing a randomly selected region of it with an unrelated monkey image. Figure 3 shows some facial images of a subject with 25% occlusions from the Extended Yale B database. We apply various clustering algorithms on the facial images of the first 2, 4, 6, 8, and 10 subjects in this database. In each test, 80% of the images of each subject are corrupted with the contiguous occlusion above.

Figure 4 shows the clustering accuracy of each method as a function of the number of subjects on the Extended Yale B database, averaged over 10 random runs. To make it more convincing, we also compute the mean, median, minimum and maximum clustering accuracy of each method over 10 random runs with a distinct number of subjects. Table 4 reports the detailed clustering results. As shown in Table 4, MEESSC achieves the highest clustering accuracy in most cases. We conduct another experiment to analyze the impact of the percent of images with occlusion on the clustering performance of each method. Let p_1 be the percent of images with occlusion in the experiment. Specifically, we fix the number of subjects $K = 2$ and vary p_1 from 0 to 80%. Figure 5 shows the clustering results. As it

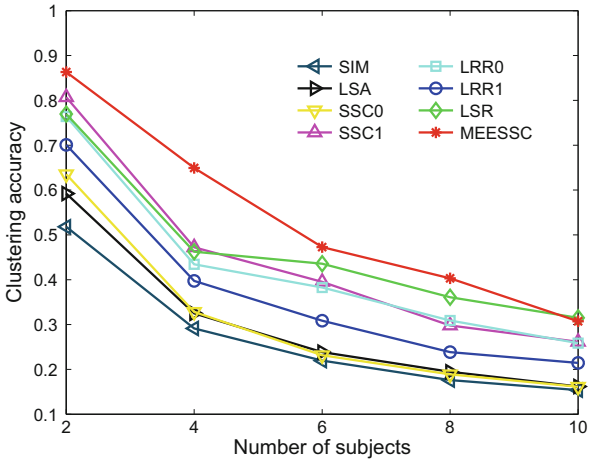


Fig. 4 Average clustering accuracy of different methods as a function of the number of subjects using ten-run test on the Extended Yale B database with 80% of images corrupted by contiguous occlusion

Table 4 Clustering accuracy of different algorithms on the Extended Yale B database with 25% random contiguous occlusion. Best results are marked bold

	Methods	SIM	LSA	SSC0	SSC1	LRR0	LRR1	LSR	MEESSC
2 Subjects	Mean	0.52	0.59	0.64	0.81	0.76	0.70	0.77	**0.86**
	Median	0.51	0.59	0.63	0.81	0.77	0.71	0.78	**0.87**
	Min	0.50	0.52	0.58	0.76	0.74	0.55	0.71	**0.84**
	Max	0.57	0.67	0.69	0.87	0.79	0.78	0.83	**0.90**
4 Subjects	Mean	0.29	0.33	0.33	0.47	0.43	0.40	0.46	**0.65**
	Median	0.29	0.33	0.33	0.46	0.42	0.39	0.46	**0.64**
	Min	0.28	0.30	0.30	0.42	0.39	0.36	0.37	**0.55**
	Max	0.30	0.37	0.35	0.61	0.53	0.47	0.60	**0.78**
6 Subjects	Mean	0.22	0.24	0.23	0.40	0.38	0.31	0.44	**0.47**
	Median	0.22	0.24	0.23	0.39	0.37	0.32	0.45	**0.47**
	Min	0.20	0.22	0.22	0.35	0.32	0.26	**0.36**	**0.36**
	Max	0.24	0.25	0.25	0.48	0.45	0.36	0.49	**0.58**
8 Subjects	Mean	0.18	0.19	0.19	0.30	0.31	0.24	0.36	**0.40**
	Median	0.17	0.20	0.19	0.29	0.31	0.23	0.36	**0.41**
	Min	0.17	0.18	0.18	0.25	0.29	0.22	**0.33**	**0.33**
	Max	0.19	0.21	0.20	0.36	0.33	0.29	0.39	**0.44**
10 Subjects	Mean	0.15	0.16	0.16	0.26	0.26	0.21	**0.31**	**0.31**
	Median	0.15	0.16	0.16	0.26	0.25	0.21	**0.32**	0.30
	Min	0.14	0.15	0.15	0.23	0.23	0.19	**0.27**	**0.27**
	Max	0.17	0.18	0.17	0.34	0.30	0.24	**0.36**	0.35

is evident, the results demonstrate the effectiveness and robustness of MEESSC in coping with the face clustering problem with contiguous occlusion. Note that LRR1 performs worse than LRR0 with lower clustering accuracy in most cases. Recall that the difference between LRR0 and LRR1 is that LRR1 uses the heuristic post-

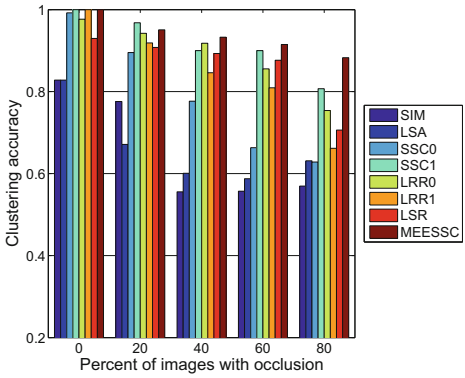


Fig. 5 Average clustering accuracy of different methods as a function of the percent of images with contiguous occlusion using ten-run test on the Extended Yale B database when $K = 2$

processing step, while LRR0 does not. This suggests that the post-processing step is not robust against the contiguous occlusion.

5.3.3 Face Clustering with Missing Entries

This subsection aims at testing the clustering performance of MEESSC on the CMU PIE database with random missing entries. Specifically, we randomly select some pixels of a facial image and replace them with zeros. Figure 3 shows some facial images with 20% missing entries from the CMU PIE database. Analogously, we apply different clustering algorithms on the facial images of the first 2, 4, 6, 8, and 10 subjects of this database. In each test, 20% pixels of each considered image are randomly selected and replaced with zeros.

Table 5 shows the clustering results of distinct methods under a varying number of subjects using ten-run test on the CMU PIE database with random missing entries. As shown in Table 5, MEESSC outperforms other competing methods in most cases. Specifically, with 20% random missing entries and $K = 2$, MEESSC achieves 98% average clustering accuracy, while the average accuracy of all other methods is less than 85%.

6 Conclusions

In this chapter, we have introduced the recent development of trajectory clustering analysis. A number of subspace-based trajectory data clustering methods have been briefly reviewed. First, we introduced a general framework called ARSC for trajectory data clustering based on atomic representation. Many state-of-the-art trajectory data clustering methods can be regarded as the specializations of ARSC with specific choices of the atomic set, loss function, and constraint. Second, by using ARSC as a general platform, we have also introduced a minimum

Table 5 Clustering accuracy of different algorithms on the CMU PIE dataset with 20% random missing entries of each image. Best results are marked bold

	Methods	SIM	LSA	SSC0	SSC1	LRR0	LRR1	LSR	MEESSC
2 Subjects	Mean	0.54	0.52	0.60	0.76	0.82	0.83	0.53	**0.98**
	Median	0.54	0.52	0.58	0.79	0.82	0.83	0.53	**0.99**
	Min	0.50	0.50	0.53	0.50	0.81	0.81	0.50	**0.96**
	Max	0.59	0.55	0.73	0.81	0.83	0.85	0.57	**1.00**
4 Subjects	Mean	0.30	0.34	0.35	0.56	0.59	0.58	0.34	**0.68**
	Median	0.29	0.33	0.36	0.57	0.59	0.58	0.34	**0.65**
	Min	0.28	0.31	0.33	0.53	**0.58**	0.56	0.31	**0.58**
	Max	0.35	0.37	0.37	0.58	0.63	0.60	0.38	**0.89**
6 Subjects	Mean	0.24	0.25	0.31	0.54	0.48	0.49	0.32	**0.60**
	Median	0.24	0.25	0.30	0.54	0.48	0.48	0.32	**0.59**
	Min	0.23	0.22	0.26	**0.50**	0.41	0.47	0.29	**0.50**
	Max	0.25	0.29	0.36	0.58	0.52	0.53	0.34	**0.77**
8 Subjects	Mean	0.19	0.22	0.30	0.44	0.42	0.43	0.27	**0.57**
	Median	0.19	0.22	0.30	0.44	0.42	0.42	0.27	**0.57**
	Min	0.18	0.20	0.28	0.38	0.36	0.41	0.25	**0.48**
	Max	0.20	0.25	0.34	0.54	0.45	0.49	0.30	**0.75**
10 Subjects	Mean	0.16	0.24	0.33	0.42	0.45	0.43	0.26	**0.53**
	Median	0.16	0.24	0.33	0.43	0.46	0.43	0.26	**0.53**
	Min	0.14	0.21	0.29	0.36	0.39	0.40	0.23	**0.45**
	Max	0.18	0.29	0.36	0.48	0.48	0.45	0.29	**0.62**

error entropy-based trajectory data clustering method referred as MEESSC. This is motivated by the sensitivity of MSE-based subspace clustering methods to non-Gaussian noise. Then we conducted comparison experiments for the real-world trajectory data clustering problems such as motion segmentation. The results validate the efficacy and robustness of MEESSC for trajectory data clustering.

References

1. J. Bian, D. Tian, Y. Tang and D. Tao, "A survey on trajectory clustering analysis," *arXiv:1802.06971*, 2018, preprint.
2. J. Costeira and T. Kanade, "A multibody factorization method for independently moving objects," *Int. J. Comput. Vis.*, vol. 29, no. 3, 1998.
3. E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 2790–2797.
4. G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 663–670.
5. G. Liu and S. Yan, "Latent low-rank representation for subspace segmentation and feature extraction," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 1615–1622.
6. Y. Zhang, Z. Sun, R. He, and T. Tan, "Robust subspace clustering via half-quadratic minimization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 3096–3103.

7. L. Wang, and M. Dong, "Detection of abnormal human behavior using a matrix approximation-based approach," in *Proc. International Conference on Machine Learning and Applications*, 2014, pp. 324–329.
8. J. Yin, Q. Yang, and J. Pan, "Sensor-based abnormal human-activity detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 8, pp. 1082–1090, Dec. 2008.
9. W. Zhao, Z. Zhang, and K. Huang, "Gestalt laws based tracklets analysis for human crowd understanding," *Pattern Recognition*, vol. 75, pp. 112–127, 2018.
10. S. Atev, O. Masoud, and N. Papanikolopoulos, "Learning traffic patterns at intersections by spectral clustering of motion trajectories," in *Proc. International Conference on Intelligent Robots and Systems*, 2006, pp. 4851–4856.
11. X. Li, J. Han, J. Lee, and H. Gonzalez, "Traffic density-based discovery of hot routes in road networks," in *Proc. International Symposium on Spatial and Temporal Databases*, 2007, pp. 441–459.
12. S. Gurung, D. Lin, W. Jiang, A. Hurson and K. Huang, "Traffic information publication with privacy preservation," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, 44, 2014.
13. P. Tseng, "Nearest q-flat to m points," *J. Optimiz. Theory App.*, vol. 105, no. 1, pp. 249–252, 2000.
14. J. Ho, M. H. Yang, J. Lim, K. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2003, pp. I–11–I–18.
15. T. Zhang, A. Szlam, and G. Lerman, "Median K-flats for hybrid linear modeling with many outliers," in *Proc. IEEE Workshop on Subspace Methods*, 2009, pp. 234–241.
16. K. Kanatani, "Motion segmentation by subspace separation and model selection," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, 2001, pp. 586–591.
17. R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1–15, Dec. 2005.
18. Y. Ma, A. Yang, H. Derksen, and R. Fossum, "Estimation of subspace arrangements with applications in modeling and segmenting mixed data," *SIAM Rev.*, vol. 50, pp. 413–458, 2008.
19. M. A. Fischler and R. C. Bolles, "RANSAC random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 26, pp. 381–395, 1981.
20. S. Rao, R. Tron, R. Vidal, and Y. Ma, "Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 10, pp. 1832–1845, Otc. 2010.
21. E. Elhamifar and R. Vidal, "Sparse subspace clustering: algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, Nov. 2013.
22. G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, Jan. 2013.
23. P. Favaro, R. Vidal, and A. Ravichandran, "A closed form solution to robust subspace estimation and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 1801–1807.
24. M. Soltanolkotabi and E. J. Candès, "A geometric analysis of subspace clustering with outliers," *Ann. Stat.*, vol. 40, no. 4, pp. 2195–2238, 2012.
25. G. Chen and G. Lerman, "Spectral curvature clustering (SCC)," *Int. J. Comput. Vis.*, vol. 81, no. 3, pp. 317–330, 2009.
26. C. Lu, J. Tang, M. Lin, L. Lin, S. Yan, and Z. Lin, "Correntropy induced L2 graph for robust subspace clustering," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1801–1808.
27. V. Zografos, L. Ellis, and R. Mester, "Discriminative subspace clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 2107–2114.
28. X. Zhang, F. Sun, G. Liu, and Y. Ma, "Fast low-rank subspace segmentation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1293–1297, May 2014.

29. J. Feng, Z. Lin, H. Xu, and S. Yan, "Robust subspace segmentation with block-diagonal prior," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014.
30. H. Hu, Z. Lin, J. Feng, and J. Zhou, "Smooth representation clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014.
31. Y. Wang, Y. Tang, and L. Li, "Minimum error entropy based sparse representation for robust subspace clustering," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4010–4021, 2015.
32. A. Ng, Y. Weiss, and M. Jordan, "On spectral clustering: analysis and an algorithm," in *Proc. NIPS*, 2001, pp. 849–856.
33. V. Patel, H. Nguyen, and R. Vidal, "Latent space sparse subspace clustering," in *IEEE Int. Conf. Comput. Vis.*, Dec 2013, pp. 225–232.
34. C. Lu, J. Feng, Z. Lin, and S. Yan, "Correlation adaptive subspace segmentation by trace lasso," in *IEEE Int. Conf. Comput. Vis.*, Dec 2013, pp. 1345–1352. i
35. R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc., Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.
36. M. Soltanolkotabi, E. Elhamifar, and E. J. Candès, "Robust subspace clustering," *Ann. Stat.*, vol. 42, no. 2, pp. 669–699, 2014.
37. Y. Wang and H. Xu, "Noisy sparse subspace clustering," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 89–97.
38. J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," in *Proc. European Conf. Comput. Vis.*, 2006, pp. 94–106.
39. T. Zhang, A. Szlam, Y. Wang, and G. Lerman, "Hybrid linear modeling via local best-fit flats," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1927–1934.
40. A. Goh and R. Vidal, "Segmenting motions of different types by unsupervised manifold clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–6.
41. A. Tewari, P. Ravikumar, and I. Dhillon, "Greedy algorithms for structurally constrained high dimensional problems," in *Proc. NIPS*, 2011, pp. 882–890.
42. W. Liu, P. P. Pokharel, and J. C. Principe, "Correntropy: properties and applications in non-Gaussian signal processing," *IEEE Trans. Signal Process.*, vol. 55, no. 11, pp. 5286–5298, Nov. 2007.
43. V. Chandrasekaran, B. Recht, P. Parrilo, and A. Willsky, "The convex geometry of linear inverse problems," *Found. Comput. Math.*, vol. 12, pp. 805–849, Dec. 2012.
44. B. Bhaskar and B. Recht, "Atomic norm denoising with applications to line spectral estimation," in *Proc. 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2011, pp. 261–268.
45. B. Trevor, T. Hadtie, L. Johnstone, and R.Tibshirani, "Least angle regression," *Ann. Stat.*, vol. 32, no. 1, pp. 407–499, 2002.
46. B. Recht, W. Xu, and B. Hassibi, "Necessary and sufficient conditions for success of the nuclear norm heuristic for rank minimization," in *Proc. IEEE Conf. Decis. Contr.*, 2008, pp. 3065–3070.
47. J. Wright, A.Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 2, pp. 210–227, Jan. 2009.
48. Y. Wang, Y. Tang, and L. Li, "Minimum Error Entropy Based Sparse Representation for Robust Subspace Clustering," *IEEE Trans. Signal Process.*, vol. 63, no. 15, pp. 4010–4021, Aug. 2015.
49. N. Rao, B. Recht, and R. Nowak, "Tight measurement bounds for exact recovery of structured sparse signals," *arXiv:1106.4355v3*, 2011, preprint.
50. C. Lu, H. Min, Z. Zhao, L. Zhu, D. Huang, and S. Yan, "Robust and efficient subspace segmentation via least squares regression," in *Proc. European Conf. Comput. Vis.*, 2012, pp. 1801–1808.
51. D. Luo, F. Nie, C. Ding, and H. Huang, "Multi-subspace representation and discovery," in *Proc. ECML PKDD*, 2011, pp. 405–420.
52. Y. Wang, H. Xu, and C. Leng, "Provable subspace clustering: when LRR meets SSC," in *Proc. NIPS*, 2013.

53. J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
54. C. E. Shannon, "A mathematical theory of communications," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.
55. A. Renyi, *Probability Theory*. New York: Elsevier, 1970.
56. E. Parzen, "On the estimation of a probability density function and the mode," *Ann. Math. Stat.*, vol. 33, no. 3, pp. 1065–1076, Sep. 1962.
57. D. Erdogmus and J. C. Principe, "An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems," *IEEE Trans. Signal Process.*, vol. 50, no. 7, pp. 1780–1786, Jul. 2002.
58. J. C. Principe, *Information theoretic learning: Renyi's entropy and kernel perspectives*. New York: Springer, 2010.
59. H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," in *Proc. NIPS*, 2007, pp. 801–808.
60. M. Nikolova and M. K. Ng, "Analysis of half-quadratic minimization methods for signal and image recovery," *SIAM J. Sci. Comput.*, vol. 27, no. 3, pp. 937–966, 2005.
61. F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Optimization with sparsity-inducing penalties," *Foundations and Trends in Machine Learning*, vol. 4, no. 1, pp. 1–106, 2011.
62. T. Cai, L. Wang, and G. Xu, "Shifting inequality and recovery of sparse signals," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1300–1308, Mar. 2010.
63. R. Tron and R. Vidal, "A benchmark for the comparison of 3-d motion segmentation algorithms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007.
64. K. Lee, J. Ho, and D. Driegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, May 2005.
65. T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression database," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1615–1618, Dec. 2003.

Clustering High-Dimensional Data

Michael E. Houle, Marie Kiermeier, and Arthur Zimek

1 Introduction

"Cluster analysis is the formal study of algorithms and methods for grouping, or classifying, objects." This definition of cluster analysis was formulated more than 30 years ago by Jain and Dubes in their classic textbook [47]. Since then, cluster analysis has become more and more challenging and specialized. In particular, the trend to collect more and more data without knowing beforehand what exactly to look for in the mass of information leads often not only to larger data sets, but also to increase in the size of the underlying attribute set. Growth in the number of attributes, or data dimensionality, is generally associated with an increase in noise—attributes that are meaningless for or even detrimental to the data analysis task at hand. As the number of attributes rises, complex and previously unknown relationships among data (such as correlations) also tend to emerge.

The analysis of high-dimensional data requires special algorithms and methods for clustering. Such algorithms do not seek clusters in the full-dimensional space, but within subspaces. In general, this cannot be equivalently solved through dimensionality reduction or feature selection followed by standard clustering algorithms, since the initial feature selection may eliminate certain subspaces that are relevant

M. E. Houle
National Institute of Informatics, Tokyo, Japan
e-mail: meh@nii.ac.jp

M. Kiermeier
Ludwig-Maximilians-Universität München, Munich, Germany
e-mail: marie.kiermeier@ifi.lmu.de

A. Zimek (✉)
Department of Mathematics and Computer Science, University of Southern Denmark, Odense M, Denmark
e-mail: zimek@imada.sdu.dk

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_11

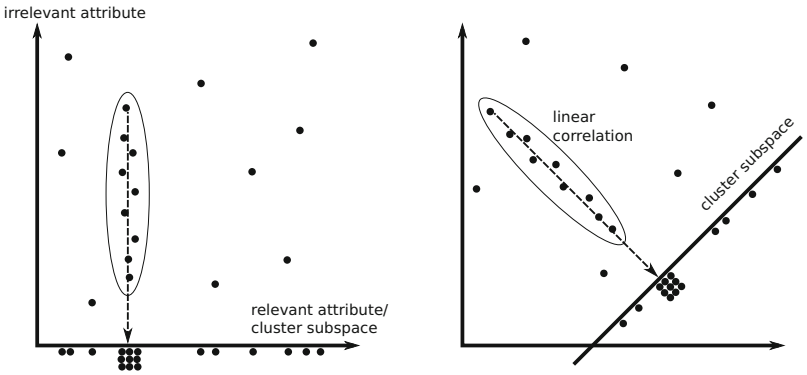


Fig. 1 Axis-parallel subspace clustering vs. arbitrarily oriented subspace clustering

to important clusters. At a high level, the task could therefore be described as that of identifying subsets of data points and a corresponding subspace for each, such that the data subset forms a cluster *within its subspace*. Figure 1 illustrates how data points forming a hyperplane in the data space could cluster densely if projected orthogonally onto a hyperplane. We also see here that we have two fundamental categories of cluster subspaces, namely *axis-parallel* or *arbitrarily oriented*, depending on the relationship of the subspace with the underlying coordinate system. These two categories come with different challenges:

1. **Axis-Parallel Subspace Clustering:**
   Since the number of potential subspaces determined by a set of attributes increases exponentially with the number of attributes (the dimension of the data set), it is not efficient to examine every possible subspace for clusters. For this reason, clustering algorithms within this category use heuristic search strategies to identify suitable axis-parallel subspaces. Essentially, the task reduces to that of distinguishing *irrelevant attributes* from those *relevant* to the formation of the cluster subspace. For example, if the projection of data points on a given axis has low variance (that is, the points are dense after projection), the attribute is considered to be relevant for clustering. Attributes are irrelevant if the projection of the data points on these axes exhibits high variance.
   It should be noted, however, that relevance or irrelevance of an attribute is not necessarily a global property, but often a local property in that it could be relevant for some cluster and irrelevant for others.
2. **Arbitrarily oriented Subspace Clustering:**
   When generalizing the idea of subspace clustering to include the identification of arbitrarily oriented cluster subspaces, the number of potential subspaces becomes infinite. Instead of determining the relevance of the uncountably many subspaces, clustering algorithms explicitly seek data subsets forming clusters after projection to some arbitrarily oriented hyperplane. Since such data subsets must be linearly correlated in the direction of the normal vector to the

hyperplane, the problem of arbitrarily oriented subspace clustering is equivalent to that of finding *linearly correlated* data subsets. Accordingly, this category of clustering is also called *correlation clustering*, and the resulting clusters are referred to as *correlation clusters*[1] and can be described by quantitative models capturing the potentially complex linear correlations [2].

In the remainder of this chapter, we survey the basic algorithmic strategies, principles, and techniques used by algorithms of these two categories, axis-parallel subspace clustering (Sect. 2) and arbitrarily oriented subspace clustering (Sect. 3). We conclude with a summary, recommendations for further reading, and a short discussion of future research directions (Sect. 4).

2 Axis-Parallel Subspace Clustering

For axis-parallel subspace clustering, we distinguish high-level subspace search strategies and cluster criteria.

2.1 Subspace Search Strategy

When restricting the clustering problem to axis-parallel subspaces, there are two fundamental search strategies for identifying relevant attribute subsets: *bottom-up* vs. *top-down*.

In top-down search, data clusters are precomputed with respect to the full attribute sets, from which irrelevant attributes are iteratively removed. A common alternative to data cluster precomputation is to use data subsets determined by local regions (such as neighborhoods) within the full-dimensional data domain. Examples of top-down axis-parallel subspace clustering methods are PROCLUS [11], FINDIT [74], SSPC [75], and PreDeCon [26], each of which adopts different clustering criteria for the search for appropriate subspaces. Many soft projected clustering methods [31, 36, 46, 48, 25, 32, 61] are variants of k-means or EM-type clustering.

In bottom-up search, individual attributes are tested for their potential relevance, yielding candidates for relevant subsets consisting of a single attribute. Thereafter, larger relevant candidate subsets are generated by combining smaller candidate subsets and testing them for relevance. To manage combinatorial explosion, the bottom-up search strategy typically relies on a heuristic from the Apriori algorithm [14] to exclude irrelevant attribute combinations. The Apriori heuristic relies on a frequency criterion that satisfies an inclusion property whereby combinations that fail to satisfy the criterion are guaranteed not to be subsets of any larger

[1] This should not be confused with the different meaning of "correlation clustering" as introduced by Bansal et al. [21].

combinations that do satisfy the criterion. We can conclude that, if there are no dense accumulations of points in some attribute combination (or within some region with respect to that combination), this combination of attributes cannot be part of a subspace within which a dense cluster can reside. With bottom-up search, management of the interaction of cluster criterion and subspace search is generally more challenging than that with top-down search.

2.2 Cluster Criteria

Examples for top-down search have been mentioned above. The interaction between subspace search and cluster criterion is typically less complicated in top-down variants, but they typically assume that the full-dimensional neighborhood of a point or the full-dimensional proto-cluster is relevant enough to identify a subspace for the corresponding cluster.

Different cluster criteria have been developed that allow the application of the Apriori principle in bottom-up clustering.

In a *grid-based* strategy, the idea is to discretize the data space into equal-sized grid cells and to characterize them by the number of points they contain, which is effectively a local density estimate. Clusters and corresponding subspaces are then found by merging dense grid cells. A visualization is presented in Fig. 2.

The figure also illustrates how Apriori-style search is adapted for axis-aligned subspace cluster determination: $(n+1)$-dimensional intersections of n-dimensional dense grid cells are considered as dense cluster *candidates* but are pruned if they do

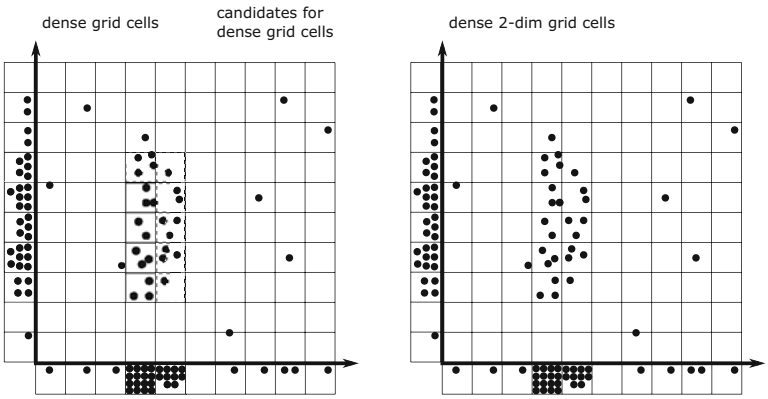


Fig. 2 Using subspace grids for finding subspace clusters: (left) 1-D dense grid cells are identified, and neighboring dense cells are merged to form 1-D clusters; (right) the intersections of 1-D dense grid cells are tested against density criteria after intersection in the combined 2-D subspace—those that satisfy the criterion (e.g., minimum density of 3 points per unit) can be merged to form 2-D clusters, and possibly joined with dense grid cells in adjacent subspaces

not meet the density criterion. Intersections involving an n-dimensional grid cell that does not satisfy the density criterion cannot in turn satisfy the $(n + 1)$-dimensional density criterion and are not checked and typically not even instantiated. Candidates (i.e., $(n + 1)$-dimensional grid cells) that qualify as dense are possibly merged with neighboring grid cells to form a cluster in that subspace. Also they are joined with intersecting dense grid cells from other $(n + 1)$-dimensional subspaces that have n attributes in common to form candidates for the corresponding $(n + 2)$-dimensional superspace of both $(n + 1)$-dimensional subspaces.

Various algorithms implement this Apriori-style search heuristic or variants thereof. The seminal method was CLIQUE [15], with well-known variants including ENCLUS [29], MAFIA [66], XProj [13], EDSC [18], and INSCY [19].

One main difficulty in applying Apriori-style methods in subspace clustering is the selection of an effective grid resolution. Given an interval of data values $[a, b]$ within which n data points reside, scale heuristics such as Sturges' rule can be used to determine the number of equal-width grid cells into which to divide the interval $\widehat{k}$, as well as the bandwidth of these cells h [70]:

$$\widehat{k} = 1 + \log_2(n)$$
$$h = \frac{b - a}{k}.$$

However, such partition heuristics do not guarantee that the resulting clusters are approximated accurately by the (merged) cells. In addition, a density threshold is required to identify dense cells. Even if the inadequacy of a global density threshold can be resolved using adaptive density thresholds, such approaches would entail an increase in the computational cost. Another weak point of the use of grid cells is their computational inefficiency, particularly for very sparse data sets or data sets of high dimensionality, where they tend to produce very large numbers of regions containing very few points (or more commonly, no points at all).

Density-based clustering methods [28] avoid the problems of discretization by assessing the density around each point with respect to subspaces of interest, provided that the criterion satisfies an Apriori-style inclusion property. Consider the simple illustration in Fig. 3. In this example, the density at a given point is assessed according to the number of neighboring points within a fixed radius, indicated by circles in the combined space of attributes A1 and A2. Clusters are determined by successively aggregating points that contain each other in their respective neighborhoods. In the left-hand figure, the cluster in the combined space in each attribute remains intact even after projection to A1 or A2, assuming that the same radius bound is used for the density criterion in the projection space (A1 or A2). In the right-hand figure, the points form a cluster with respect to A1; however, with respect to A2, the point with greatest value of A2 does not belong to the density-based cluster in this attribute. This point therefore cannot belong to a density-based cluster (using the same thresholds or criteria) in any superspace of A2 regardless of its participation in clusters in other spaces (such as A1 or its

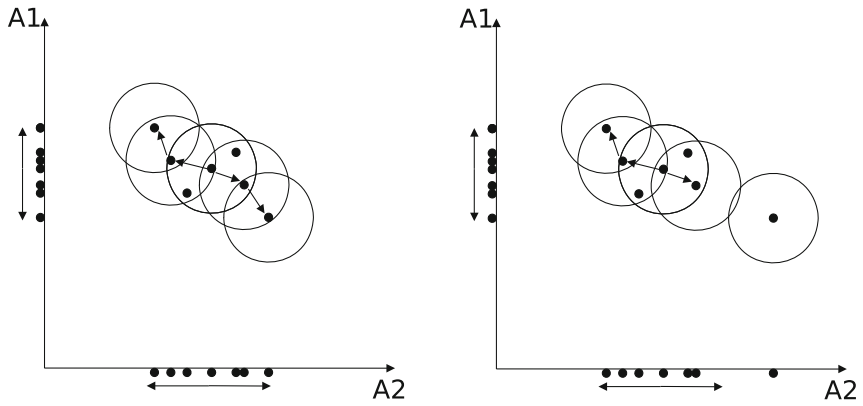


Fig. 3 Using density-based concepts for finding subspace clusters: (left) points forming a density-based cluster in the combination of attributes A1 and A2 would also form density-based clusters in each attribute individually when using the same density threshold; (right) even though the rightmost point belongs to the density-based cluster in A1, it does not belong to the density-based cluster in the space formed by A1 and A2 due to the gap in its A2 value relative to the other points

superspaces). As a consequence, if there is no density-based cluster at all with respect to some subspace (a combination of attributes), all of its superspaces can be ruled out for further search.

Examples of subspace clustering methods using density-based criteria include SUBCLU [50] and DUSC [17].

Finally, the Apriori principle (or related search strategies) can also be used in connection with variance in local neighborhoods. Examples for such approaches include CFPC [76], HiCS [1], and DiSH [4].

3 Arbitrarily Oriented Subspace Clustering

The techniques used for axis-parallel subspace clustering are generally straightforward, with only minor differences among them in their various implementations. In contrast, the techniques used for the more challenging problem of arbitrarily oriented subspace search are considerably more diverse, although some have seen only limited use.

3.1 Principal Component Analysis (PCA)

The idea of principal component analysis (PCA) is to describe the structure of a set of data points with respect to a transformation to a new coordinate system, where

the basis vectors, the *principal components* (PCs), are ordered according to the amount of variance in the data with respect to their directions [49]. Given a vector $x = (x_1, x_2, \ldots, x_p)^\intercal$ of p random variables, the first PC z_1 is a vector $\alpha_1 = (\alpha_{11}, \alpha_{12}, \ldots, \alpha_{1p})^\intercal$ of p constants such that the following linear combinations have maximum variance when taken over the data set:

$$z_1 = (\alpha_1)^\intercal x = \alpha_{11}x_1 + \alpha_{12}x_2 + \cdots + \alpha_{1p}x_p = \sum_{j=1}^{p} \alpha_{1j}x_j.$$

The second PC z_2 is the linear function $(\alpha_2)^\intercal x$ with maximum variance over all possibilities that are orthogonal to α_1—that is, such that $\alpha_2 \cdot \alpha_1 = 0$. Similarly, successive PCs of the form $z_k = (\alpha_k)^\intercal x$ capture the maximum variance over all possibilities that are orthogonal to α_i, for all $1 \leq i < k$.

The computation of the PCs is typically based on the covariance matrix Σ of x. The k-th PC is thus given by $z_k = \alpha_k^\intercal x$, where α_k is the eigenvector of the k-th largest eigenvalue λ_k of the covariance matrix Σ. If $\alpha_1, \alpha_2, \ldots, \alpha_p$ are normalized ($\alpha_k \in [0, 1]$ for $i = 1, 2, \ldots, p$), the eigenvalue λ_k is precisely the variance of the data points from the mean along its corresponding eigenvector z_k. The eigenvectors form an orthogonal system (called an *eigensystem*), which span the original data space, but in directions for which most of the spread is associated with the first few coordinates.

Removing the PCs with smallest eigenvalues (those with the least variance and carrying the least information) can result in an arbitrarily oriented subspace that describes the data with only minimal loss of information. Such subspaces correspond to the hyperplanes constructed through data point correlation. Accordingly, to define the orthogonal (complementary) subspace in which the data points cluster densely, the PCs with the greatest eigenvalues are omitted, and only those associated with the smallest eigenvalues are taken into account (see Fig. 4).

The majority of existing algorithms use PCA to identify arbitrarily oriented subspaces. The earliest example, ORCLUS [10], employs a variant of the k-means partitioning algorithm, in which every partition is represented by its centroid and eigensystem. An initial data partition is constructed using an initial set of randomly

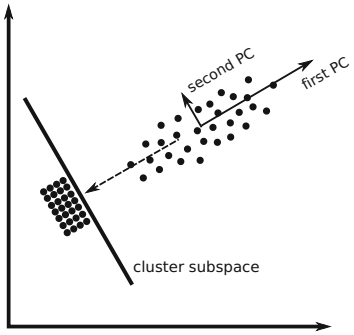


Fig. 4 Using PCA for the identification of subspace clusters. Eigenvectors with low variance define a subspace capturing a dense clustering of points

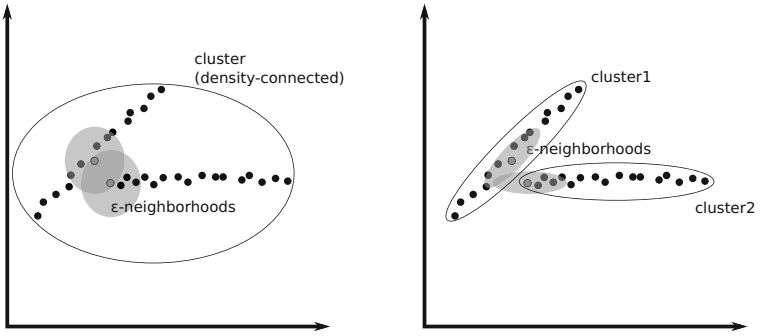


Fig. 5 ε-neighborhood (left) using Euclidean distance and (right) using an eigensystem-based distance

chosen data points as centroids. In each iteration, the dimensions of the eigensystems are successively reduced by removing the PCs with greatest eigenvalues. The data points are then (re)assigned to the partitions so as to minimize the average distance of the points to the centroid after projection to the subspace spanned by the current set of PCs. The algorithm terminates when the subspace dimension reaches a predefined value. Variants of this strategy include DPCLUS [59] and ROSECC [20].

4C [27] extends the density-based approach of the classic DBSCAN clustering algorithm [35] by using a locally adaptive distance measure based on the eigensystems of each point (see Fig. 5). Variants of 4C include COPAC [6] as well as hierarchical methods such as HiCO [3] and ERiC [5], which summarize the hierarchical inclusion relationships among subspaces. Some algorithms, such as CARE [78] and SSCC [40], do not apply PCA in the full-dimension data space, but in heuristically selected attribute subspaces. Variants of PCA more resilient to noise have been proposed for use with correlation clustering algorithms [54, 53].

Generalized variants of PCA [73] and regression [72] led to numerous related approaches to subspace clustering. Some of the more prominent examples are sparse subspace clustering [34, 82] and low-rank subspace clustering [60].

3.2 Grid-Based Cluster Identification

To find arbitrarily oriented subspace clusters, algorithms of this group first discretize the attribute spaces into grid cells and then iteratively combine attributes to form candidate subspaces. Arbitrarily oriented subspace clusters are then detected using suitable merging strategies (see Fig. 6). If only subspaces of dimension one are used—that is, projections on original attributes—this approach corresponds to the grid-based approach to axis-parallel subspace clustering.

Examples following this strategy are EPCH [67] and P3C [62, 63]. In EPCH, to identify d-dimensional dense grid cells, every $1, 2, \ldots, d$-dimensional subspace

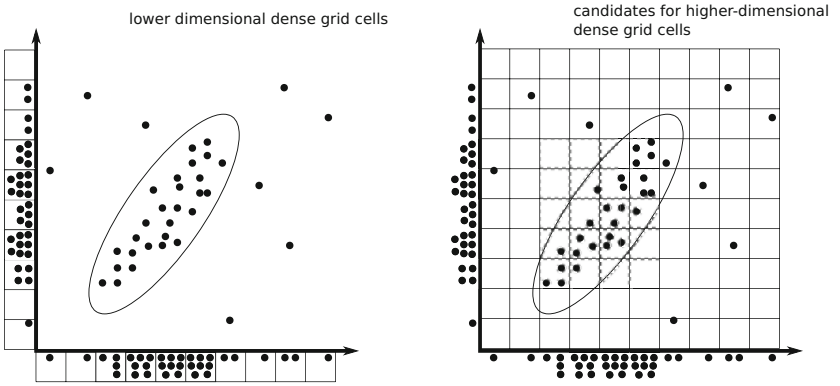


Fig. 6 Subspace grids for finding arbitrarily oriented subspace clusters

that can be formed from d attributes is discretized into uniformly spaced cells. Each data point is then represented by a signature that lists all subspaces and corresponding dense grid cells that contain the point. To find subspace clusters, suitable cells are merged by comparing their signatures.

Since EPCH employs an adaptive density threshold for identifying dense cells, it does not suffer from the inaccuracy of a user-defined global density threshold. Only one density parameter needs to be set by the user. However, the time complexity can be seen to increase as $O(d^D)$, where D denotes the dimension of the data set. Although EPCH has been shown to be experimentally effective in certain settings for the choices of $d = 1$ (known as the EPC1 variant) and $d = 2$ (the EPC2 variant), in general a satisfactory tradeoff between accuracy (obtained by increasing d) and runtime (lowered by decreasing d) may not be possible to obtain.

In contrast with EPCH, for P3C [62, 63], the initial dimension of the subspaces is fixed at one. Accordingly, for each attribute (1-dimensional subspace), intervals that are not uniformly distributed are identified using the chi-square goodness-of-fit test. By combining such significant intervals, hyperrectangles composed of higher-dimensional grid cells are tested for density, according to whether they contain more data points than would be expected according to a Poisson probability function. Finally, all data points are assigned to dense cells by applying the EM algorithm to determine subspace cluster membership. As with EPCH, P3C uses an adaptive density threshold that requires the user to set a single parameter, for the Poisson probability function.

There are grid-based algorithms that do not use a bottom-up approach to derive subspace clusters, but instead rely solely on full-dimensional grids. Algorithms of this category inherently assume that subspace clusters cause density abnormalities even in the full-dimensional space. An example of this kind of approach is Halite [30].

3.3 Hough Transform

The basic goal of the original Hough transform [43] is to discover collinearities in data—that is, subsets of the data that can be approximated by a straight line. The Hough transform maps data points into a dual space, often referred to as the *parameter space*. The dual mapping proposed by Duda and Hart [33] uses an *angle–radius* parameterization instead of the original *slope–intercept* form. Under the angle–radius mapping, a 2-dimensional data point $p = (x, y)$ is described by all possible lines passing through this point (*parameterization function*):

$$x \cdot \cos(\theta) + y \cdot \sin(\theta) = \rho,$$

where θ denotes the angle of the normal vector of the line, which is restricted to the range $\theta \in [0, \pi]$ for uniqueness, and ρ is the distance from the origin. For each data point, the set of all lines containing it in the original space generates a dual curve in the parameter space whose points consist of pairs of the form (θ, ρ). Data points in the original space whose dual curves intersect in a common location (θ, ρ) in parameter space would be collinear with respect to the line whose angle and radius are θ and ρ, respectively. This duality principle is illustrated in Fig. 7.

For subspace clustering in higher dimensional spaces, an extended version of the Hough duality has been applied to find co-planarities of points, in CASH [8, 7]. Here, a d-dimensional data point $p = (p_1, p_2, \ldots, p_d)^\mathsf{T}$ is described by all possible hyperplanes containing p. The corresponding parametrization function is given by

$$\sum_{i=1}^{d} p_i \cdot \prod_{j=1}^{j-1} \sin(\alpha_j) \cdot \cos(\alpha_i) = \rho,$$

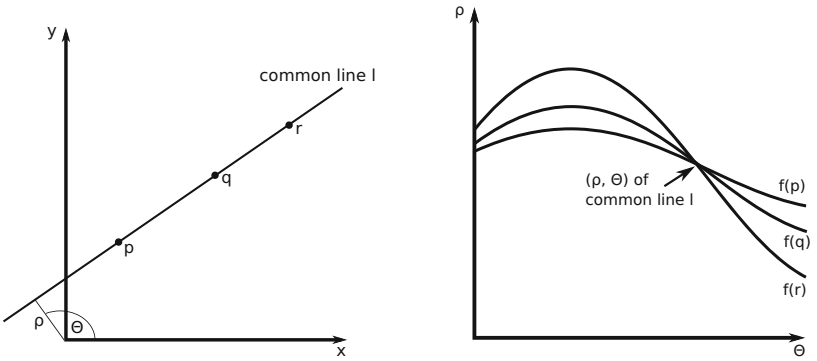


Fig. 7 Hough transform: data space (left) and parameter space (right)

where $\alpha_1, \alpha_2, \ldots \alpha_{d-1}$ are $d - 1$ angles of the normal vectors defining the hyperplanes in Hessian normal form. Again, each data point is represented in the parameter space by its parameterization function, and an intersection of the parameterization functions implies that the corresponding data points lie on a common hyperplane in the original data space. The algorithm searches recursively for lower-dimensional co-planarities within higher-dimensional hyperplanes.

Recent work following up on this technique focused on faster search strategies in the parameter space [52] or the identification of non-linear dependencies [51].

3.4 Random Sampling

The Hough transform method is limited to the detection of co-planarities of dimension one less than that of the current space. To find subspaces of smaller dimensions requires an iterative application, descending through the dimensionalities one by one. A non-exhaustive yet more direct alternative is to use random sampling methods to generate candidate subspaces for further testing, using the more general concept of a *linear manifold*.

Linear manifolds can be regarded as translated subspaces without an origin such that non-homogeneous linear combinations of attributes are allowed. Their general form can be described as

$$p = \mu + B * \lambda,$$

where μ is a fixed translation vector (determining an origin relative to the manifold), B is a matrix of a subset of orthonormal vectors spanning the subspace, and λ is a vector determining the position of p within the manifold.

In order to fit a linear manifold to a set of data points $p_1, p_2, \ldots, p_n$, we can extend the usual representation to account for points that may be nearby but not contained in it [41, 42]:

$$p_i = \mu + B * \lambda_i + \overline{B} * \psi_i,$$

where $\overline{B}$ is a matrix of orthonormal basis vectors that extends B so as to cover the full-dimensional space, and ψ_i is a vector determining the offset of p_i from the manifold described by B and μ. For the manifold to be a good fit, in any given coordinate of the vectors ψ_i, the entries should have low variance across the n points, as compared to the variance within the coordinates of the vectors λ_i.

With this in mind, the random sampling strategy identifies correlated data points located on or near linear manifolds of a target dimension, by randomly sampling data points to form an orthonormal basis B, as follows [41, 42]:

- For a d-dimensional manifold, $d + 1$ linearly independent data points $x_0, x_1, \ldots, x_d$ are randomly sampled from the data set.

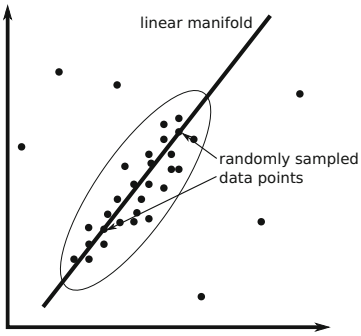


Fig. 8 Using randomly sampled linear manifolds for finding subspace clusters

- x_0 is chosen to be the center.
- The vectors $y_j = x_j - x_0$ (for $j = 1, 2, \ldots, d$) are used to construct B by applying the Gram–Schmidt method.

If a randomly generated manifold can be used to capture part of the data (with low variance across the offset vectors ψ_i), the corresponding data points are deemed to form a subspace cluster (see Fig. 8).

In contrast to the previously presented strategies, this approach avoids an explicit analysis of the data distribution. The random sampling process precludes a deterministic result, nor does it guarantee good coverage. In addition, the time complexity increases exponentially with the maximum dimension of the sampled manifolds. This strategy is therefore viable only for subspace clusters of low dimensionality.

3.5 Intrinsic Dimensionality

Finally, the estimation of the *intrinsic dimensionality* of cluster candidates according to some model has been used for clustering high-dimensional data. Collectively, models of intrinsic dimensionality seek to identify the numbers of latent features (dimensions) that best characterize subsets of interest, irrespective of the actual data representation being used, and without necessarily constructing a subspace embedding. Dimensionality-aware subspace clustering algorithms typically rely on the model to determine groups of points for which the intrinsic dimension is relatively low, and are thus attracted to groups with simpler explanations in terms of latent features. For a discussion and experimental comparison of various models of intrinsic dimensionality, see, for example, the work of Amsaleg et al. [16].

Fractal clustering (FC) [22] constructs an initial set of cluster prototypes from an initial random subset of the data, by using a nearest-neighbor-based chaining process. Thereafter, the remaining data points are added to clusters one by one, choosing the cluster for which the addition of the new point would result in the smallest change in the cluster's intrinsic dimensionality. If the change would exceed

a certain threshold, the addition of the point is not performed, and the point is declared to be noise.

FC is capable of finding even non-linearly correlated subspace clusters. However, one major drawback of FC is its dependence on the clustering of the initial random sample. If the initialization fails to capture the basic subspace structure of the data, the subsequent clustering process cannot recover.

In the original implementation, the Hausdorff dimension is used for the estimation of intrinsic dimensionality over a coarsened representation of the cluster by multiresolutional grid storing counts of points.

In contrast to FC, dimension-induced clustering (DIC) [38] computes intrinsic dimensional information for every point of the data. Given a point v, two quantities are estimated over the set of points contained in a neighborhood of v: the correlation dimension and the density. These values effectively transform the data into a 2-dimensional setting within which expectation maximization (EM) is used to find clusters. The members of these clusters determine the final subspace clusters.

Like FC, DIC is capable of finding subspace clusters that are not linearly correlated. Unlike FC, hierarchies of subclusters can also be detected. In addition, by mapping the data into a 2-dimensional setting, the computational problems associated with high-dimensional clustering are avoided. However, DIC must be supplied with two parameters that have a critical impact on the composition of the final clustering: the number of clusters sought (required by the EM algorithm) and the neighborhood radius (for the calculation of the correlation dimension and local density).

4 Conclusion

This chapter presented an overview of the basic strategies and techniques used for subspace clustering in high-dimensional settings, with a high-level distinction made between those that restrict clusters to axis-parallel subspaces and those that allow the subspaces to be arbitrarily oriented. For each category, several strategies have been explored, and examples of methods provided.

4.1 Further Reading

There are several surveys available on clustering high-dimensional data, some broader, and others focusing on specific aspects. A broad and extensive survey has been provided by Kriegel et al. [55], categorizing according to algorithmic strategies and variants of the problem definition. They also present the relationship between subspace clustering and pattern-based or bi-clustering approaches. More concise surveys have also been published by the same authors [58, 79]. Vidal et al. [73] give an overview of variants of PCA and PCA-based subspace clustering

methods targeting applications in image processing and computer vision. Sim et al. [71] discuss more challenging variations of the subspace clustering problem. The relation of certain algorithmic approaches to the ideas developed in frequent pattern mining has been discussed by Zimek et al. [81]. Zimek and Vreeken [80] explored the relationships between subspace clustering and ensemble clustering, alternative clustering, and multiview clustering. Evaluation studies have been provided by Müller et al. [65] and Moise et al. [64].

4.2 Future Research Directions

The current state of the art of clustering high-dimensional data suggests future research in three aspects:

1. **Primitives to capture subspaces.** While PCA has seen substantial use in determining local subspaces in data, it is computationally heavy in that it necessitates the computation of the local subspace itself. This is in contrast with dimension-induced clustering (DIC), which requires only an estimate of the local intrinsic dimensionality. To date, the use of local measures of intrinsic dimensionality to aggregate points without explicitly computing an embedding is relatively unexplored. Such methods would also depend greatly on the characteristics of the models of intrinsic dimensionality used, as well as their associated estimators. Recent work has focused on the use of local intrinsic dimensionality (LID) [44, 45] to identify subspace dimension [23]. The exploration of further primitives based on intrinsic dimensionality can be expected to yield interesting new possibilities for subspace detection.
2. **Dynamic and streaming data.** Extending the clustering of high-dimensional data to scenarios with dynamic or streaming data comes with additional challenges. Work in this area is still in its early stages—examples include [12, 77, 9, 56, 68, 24].
3. **Evaluation.** The evaluation of unsupervised learning is notoriously challenging. Additional consideration of the quality of exact or approximate subspace determination would bring an added layer of complexity to the clustering evaluation problem. For correlation clusters, a quantitative model can be derived [2] that could also be used in a predictive evaluation scenario. In the literature, only a handful of evaluation studies [69, 65, 64, 39] have addressed the issue of subspace clustering. In scenarios in which cluster overlap is allowed—that is, when a given point can belong to different clusters in different subspaces—subspace clustering would have many aspects in common with multiview clustering or alternative clustering [80], adding to the challenges surrounding evaluation [37, 57].

References

[1] Achtert E, Böhm C, Kriegel HP, Kröger P, Müller-Gorman I, Zimek A (2006) Finding hierarchies of subspace clusters. In: Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Berlin, Germany, pp 446–453, https://doi.org/10.1007/11871637_42

[2] Achtert E, Böhm C, Kriegel HP, Kröger P, Zimek A (2006) Deriving quantitative models for correlation clusters. In: Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Philadelphia, PA, pp 4–13, https://doi.org/10.1145/1150402.1150408

[3] Achtert E, Böhm C, Kröger P, Zimek A (2006) Mining hierarchies of correlation clusters. In: Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM), Vienna, Austria, pp 119–128, https://doi.org/10.1109/SSDBM.2006.35

[4] Achtert E, Böhm C, Kriegel HP, Kröger P, Müller-Gorman I, Zimek A (2007) Detection and visualization of subspace cluster hierarchies. In: Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA), Bangkok, Thailand, pp 152–163, https://doi.org/10.1007/978-3-540-71703-4_15

[5] Achtert E, Böhm C, Kriegel HP, Kröger P, Zimek A (2007) On exploring complex relationships of correlation clusters. In: Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM), Banff, Canada, pp 7–16, https://doi.org/10.1109/SSDBM.2007.21

[6] Achtert E, Böhm C, Kriegel HP, Kröger P, Zimek A (2007) Robust, complete, and efficient correlation clustering. In: Proceedings of the 7th SIAM International Conference on Data Mining (SDM), Minneapolis, MN, pp 413–418, https://doi.org/10.1137/1.9781611972771.37

[7] Achtert E, Böhm C, David J, Kröger P, Zimek A (2008) Global correlation clustering based on the Hough transform. Statistical Analysis and Data Mining 1(3):111–127, https://doi.org/10.1002/sam.10012

[8] Achtert E, Böhm C, David J, Kröger P, Zimek A (2008) Robust clustering in arbitrarily oriented subspaces. In: Proceedings of the 8th SIAM International Conference on Data Mining (SDM), Atlanta, GA, pp 763–774, https://doi.org/10.1137/1.9781611972788.69

[9] Aggarwal CC (2009) On high dimensional projected clustering of uncertain data streams. In: Proceedings of the 25th International Conference on Data Engineering (ICDE), Shanghai, China, pp 1152–1154, https://doi.org/10.1109/ICDE.2009.188

[10] Aggarwal CC, Yu PS (2000) Finding generalized projected clusters in high dimensional space. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX, pp 70–81, https://doi.org/10.1145/342009.335383

[11] Aggarwal CC, Procopiuc CM, Wolf JL, Yu PS, Park JS (1999) Fast algorithms for projected clustering. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), Philadelphia, PA, pp 61–72, https://doi.org/10.1145/304182.304188

[12] Aggarwal CC, Han J, Wang J, Yu PS (2005) On high dimensional projected clustering of data streams. Data Mining and Knowledge Discovery 10:251–273, https://doi.org/10.1007/s10618-005-0645-7

[13] Aggarwal CC, Ta N, Wang J, Feng J, Zaki M (2007) XProj: a framework for projected structural clustering of xml documents. In: Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Jose, CA, pp 46–55

[14] Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), Santiago de Chile, Chile, pp 487–499

[15] Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), Seattle, WA, pp 94–105, https://doi.org/10.1145/276304.276314

[16] Amsaleg L, Chelly O, Furon T, Girard S, Houle ME, Kawarabayashi K, Nett M (2018) Extreme-value-theoretic estimation of local intrinsic dimensionality. Data Mining and Knowledge Discovery 32(6):1768–1805
[17] Assent I, Krieger R, Müller E, Seidl T (2007) DUSC: dimensionality unbiased subspace clustering. In: Proceedings of the 7th IEEE International Conference on Data Mining (ICDM), Omaha, NE, pp 409–414, https://doi.org/10.1109/ICDM.2007.49
[18] Assent I, Krieger R, Müller E, Seidl T (2008) EDSC: efficient density-based subspace clustering. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM), Napa Valley, CA, pp 1093–1102, https://doi.org/10.1145/1458082.1458227
[19] Assent I, Krieger R, Müller E, Seidl T (2008) INSCY: indexing subspace clusters with in-process-removal of redundancy. In: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM), Pisa, Italy, pp 719–724, https://doi.org/10.1109/ICDM.2008.46
[20] Aziz MS, Reddy CK (2010) A robust seedless algorithm for correlation clustering. In: Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Hyderabad, India, pp 28–37
[21] Bansal N, Blum A, Chawla S (2004) Correlation clustering. Machine Learning 56:89–113, https://doi.org/10.1023/B:MACH.0000033116.57574.95
[22] Barbará D, Chen P (2000) Using the fractal dimension to cluster datasets. In: Proceedings of the 6th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Boston, MA, pp 260–264, https://doi.org/10.1145/347090.347145
[23] Becker R, Hafnaoui I, Houle ME, Li P, Zimek A (2019) Subspace determination through local intrinsic dimensional decomposition. In: Proceedings of the 12th International Conference on Similarity Search and Applications (SISAP), Newark, NJ, pp 281–289
[24] Borutta F, Kröger P, Hubauer T (2019) A generic summary structure for arbitrarily oriented subspace clustering in data streams. In: Proceedings of the 12th International Conference on Similarity Search and Applications (SISAP), Newark, NJ, pp 203–211
[25] Bouveyron C, Girard S, Schmid C (2007) High-dimensional data clustering. Computational Statistics and Data Analysis 52:502–519, https://doi.org/10.1016/j.csda.2007.02.009
[26] Böhm C, Kailing K, Kriegel HP, Kröger P (2004) Density connected clustering with local subspace preferences. In: Proceedings of the 4th IEEE International Conference on Data Mining (ICDM), Brighton, UK, pp 27–34, https://doi.org/10.1109/ICDM.2004.10087
[27] Böhm C, Kailing K, Kröger P, Zimek A (2004) Computing clusters of correlation connected objects. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), Paris, France, pp 455–466, https://doi.org/10.1145/1007568.1007620
[28] Campello RJGB, Kröger P, Sander J, Zimek A (2020) Density-based clustering. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 10(2), https://doi.org/10.1002/widm.1343
[29] Cheng CH, Fu AWC, Zhang Y (1999) Entropy-based subspace clustering for mining numerical data. In: Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Diego, CA, pp 84–93, https://doi.org/10.1145/312129.312199
[30] Cordeiro RLF, Traina AJM, Faloutsos C, Traina Jr C (2013) Halite: Fast and scalable multiresolution local-correlation clustering. IEEE Transactions on Knowledge and Data Engineering 25(2):387–401, https://doi.org/10.1109/TKDE.2011.176
[31] Domeniconi C, Papadopoulos D, Gunopulos D, Ma S (2004) Subspace clustering of high dimensional data. In: Proceedings of the 4th SIAM International Conference on Data Mining (SDM), Lake Buena Vista, FL, https://doi.org/10.1137/1.9781611972740.58
[32] Domeniconi C, Gunopulos D, Ma S, Yan B, Al-Razgan M, Papadopoulos D (2007) Locally adaptive metrics for clustering high dimensional data. Data Mining and Knowledge Discovery 14(1):63–97, https://doi.org/10.1007/s10618-006-0060-8
[33] Duda RO, Hart PE (1972) Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM 15(1):11–15, https://doi.org/10.1145/361237.361242

[34] Elhamifar E, Vidal R (2009) Sparse subspace clustering. In: Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), Miami, FL, pp 2790–2797, https://doi.org/10.1109/CVPR.2009.5206547
[35] Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR, pp 226–231
[36] Friedman JH, Meulman JJ (2004) Clustering objects on subsets of attributes. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 66(4):825–849
[37] Färber I, Günnemann S, Kriegel HP, Kröger P, Müller E, Schubert E, Seidl T, Zimek A (2010) On using class-labels in evaluation of clusterings. In: MultiClust: 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with KDD 2010, Washington, DC
[38] Gionis A, Hinneburg A, Papadimitriou S, Tsaparas P (2005) Dimension induced clustering. In: Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Chicago, IL, https://doi.org/10.1145/1081870.1081880
[39] Günnemann S, Färber I, Müller E, Assent I, Seidl T (2011) External evaluation measures for subspace clustering. In: Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM), Glasgow, UK, pp 1363–1372, https://doi.org/10.1145/2063576.2063774
[40] Günnemann S, Färber I, Virochsiri K, Seidl T (2012) Subspace correlation clustering: finding locally correlated dimensions in subspace projections of the data. In: Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Beijing, China, pp 352–360, https://doi.org/10.1145/2339530.2339588
[41] Haralick R, Harpaz R (2005) Linear manifold clustering. In: Proceedings of the 4th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM), Leipzig, Germany, pp 132–141
[42] Haralick RM, Harpaz R (2007) Linear manifold clustering in high dimensional spaces by stochastic search. Pattern Recognition 40(10):2672–2684, https://doi.org/10.1016/j.patcog.2007.01.020
[43] Hough PVC (1962) Methods and means for recognizing complex patterns. U.S. Patent 3069654
[44] Houle ME (2017) Local intrinsic dimensionality I: an extreme-value-theoretic foundation for similarity applications. In: Proceedings of the 10th International Conference on Similarity Search and Applications (SISAP), Munich, Germany, pp 64–79
[45] Houle ME (2017) Local intrinsic dimensionality II: multivariate analysis and distributional support. In: Proceedings of the 10th International Conference on Similarity Search and Applications (SISAP), Munich, Germany, pp 80–95
[46] Huang JZ, Ng MK, Rong H, Li Z (2005) Automated variable weighting in k-means type clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(5):657–668, https://doi.org/10.1109/TPAMI.2005.95
[47] Jain AK, Dubes RC (1988) Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs
[48] Jing L, Ng MK, Huang JZ (2007) An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. IEEE Transactions on Knowledge and Data Engineering 19(8):1026–1041, https://doi.org/10.1109/TKDE.2007.1048
[49] Jolliffe IT (2002) Principal Component Analysis, 2nd edn. Springer
[50] Kailing K, Kriegel HP, Kröger P (2004) Density-connected subspace clustering for high-dimensional data. In: Proceedings of the 4th SIAM International Conference on Data Mining (SDM), Lake Buena Vista, FL, pp 246–257, https://doi.org/10.1137/1.9781611972740.23
[51] Kazempour D, Mauder M, Kröger P, Seidl T (2017) Detecting global hyperparaboloid correlated clusters based on Hough transform. pp 31:1–31:6
[52] Kazempour D, Bein K, Kröger P, Seidl T (2018) D-MASC: A novel search strategy for detecting regions of interest in linear parameter space. In: Proceedings of the 11th International Conference on Similarity Search and Applications (SISAP), Lima, Peru, pp 163–176

[53] Kazempour D, Hünemörder M, Seidl T (2019) On coMADs and Principal Component Analysis. In: Proceedings of the 12th International Conference on Similarity Search and Applications (SISAP), Newark, NJ, pp 273–280
[54] Kriegel HP, Kröger P, Schubert E, Zimek A (2008) A general framework for increasing the robustness of PCA-based correlation clustering algorithms. In: Proceedings of the 20th International Conference on Scientific and Statistical Database Management (SSDBM), Hong Kong, China, pp 418–435, https://doi.org/10.1007/978-3-540-69497-7_27
[55] Kriegel HP, Kröger P, Zimek A (2009) Clustering high dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Transactions on Knowledge Discovery from Data (TKDD) 3(1):1–58, https://doi.org/10.1145/1497577.1497578
[56] Kriegel HP, Kröger P, Ntoutsi E, Zimek A (2011) Density based subspace clustering over dynamic data. In: Proceedings of the 23rd International Conference on Scientific and Statistical Database Management (SSDBM), Portland, OR, pp 387–404, https://doi.org/10.1007/978-3-642-22351-8_24
[57] Kriegel HP, Schubert E, Zimek A (2011) Evaluation of multiple clustering solutions. In: 2nd MultiClust Workshop: Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with ECML PKDD 2011, Athens, Greece, pp 55–66
[58] Kriegel HP, Kröger P, Zimek A (2012) Subspace clustering. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(4):351–364, https://doi.org/10.1002/widm.1057
[59] Li J, Huang X, Selke C, Yong J (2007) A fast algorithm for finding correlation clusters in noise data. In: Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Nanjing, China, pp 639–647
[60] Liu G, Lin Z, Yan S, Sun J, Yu Y, Ma Y (2013) Robust recovery of subspace structures by low-rank representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(1):171–184, https://doi.org/10.1109/TPAMI.2012.88
[61] Lu Y, Wang S, Li S, Zhou C (2010) Particle swarm optimizer for variable weighting in clustering high-dimensional data. Machine Learning 82(1):43–70, https://doi.org/10.1007/s10994-009-5154-2
[62] Moise G, Sander J, Ester M (2006) P3C: A robust projected clustering algorithm. In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM), Hong Kong, China, pp 414–425, https://doi.org/10.1109/ICDM.2006.123
[63] Moise G, Sander J, Ester M (2008) Robust projected clustering. Knowledge and Information Systems (KAIS) 14(3):273–298, https://doi.org/10.1007/s10115-007-0090-6
[64] Moise G, Zimek A, Kröger P, Kriegel HP, Sander J (2009) Subspace and projected clustering: Experimental evaluation and analysis. Knowledge and Information Systems (KAIS) 21(3):299–326, https://doi.org/10.1007/s10115-009-0226-y
[65] Müller E, Günnemann S, Assent I, Seidl T (2009) Evaluating clustering in subspace projections of high dimensional data. Proceedings of the VLDB Endowment 2(1):1270–1281
[66] Nagesh HS, Goil S, Choudhary A (2001) Adaptive grids for clustering massive data sets. In: Proceedings of the 1st SIAM International Conference on Data Mining (SDM), Chicago, IL, pp 1–17, https://doi.org/10.1137/1.9781611972719.7
[67] Ng KKE, Fu AW, Wong CW (2005) Projective clustering by histograms. IEEE Transactions on Knowledge and Data Engineering 17(3):369–383, https://doi.org/10.1109/TKDE.2005.47
[68] Ntoutsi E, Zimek A, Palpanas T, Kröger P, Kriegel HP (2012) Density-based projected clustering over high dimensional data streams. In: Proceedings of the 12th SIAM International Conference on Data Mining (SDM), Anaheim, CA, pp 987–998, https://doi.org/10.1137/1.9781611972825.85
[69] Patrikainen A, Meila M (2006) Comparing subspace clusterings. IEEE Transactions on Knowledge and Data Engineering 18(7):902–916, https://doi.org/10.1109/TKDE.2006.106
[70] Scott DW (2009) Sturges' rule. Wiley Interdisciplinary Reviews: Computational Statistics 1(3):303–306

[71] Sim K, Gopalkrishnan V, Zimek A, Cong G (2013) A survey on enhanced subspace clustering. Data Mining and Knowledge Discovery 26(2):332–397, https://doi.org/10.1007/s10618-012-0258-x

[72] Tibshirani R (1996) Regression shrinkage and selection via the lasso. J Royal Statistical Society, Series B 58(1):267–288

[73] Vidal R, Ma Y, Sastry SS (2016) Generalized Principal Component Analysis, Interdisciplinary Applied Mathematics, vol 40. Springer, https://doi.org/10.1007/978-0-387-87811-9

[74] Woo KG, Lee JH, Kim MH, Lee YJ (2004) FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting. Information and Software Technology 46(4):255–271, https://doi.org/10.1016/j.infsof.2003.07.003

[75] Yip KY, Cheung DW, Ng MK (2005) On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. In: Proceedings of the 21st International Conference on Data Engineering (ICDE), Tokyo, Japan, pp 329–340, https://doi.org/10.1109/ICDE.2005.96

[76] Yiu ML, Mamoulis N (2005) Iterative projected clustering by subspace mining. IEEE Transactions on Knowledge and Data Engineering 17(2):176–189, https://doi.org/10.1109/TKDE.2005.29

[77] Zhang Q, Liu J, Wang W (2007) Incremental subspace clustering over multiple data streams. In: Proceedings of the 7th IEEE International Conference on Data Mining (ICDM), Omaha, NE, pp 727–732, https://doi.org/10.1109/ICDM.2007.100

[78] Zhang X, Pan F, Wang W (2008) CARE: finding local linear correlations in high dimensional data. In: Proceedings of the 24th International Conference on Data Engineering (ICDE), Cancun, Mexico, pp 130–139, https://doi.org/10.1109/ICDE.2008.4497421

[79] Zimek A (2013) Clustering high-dimensional data. In: Aggarwal CC, Reddy CK (eds) Data Clustering: Algorithms and Applications, CRC Press, chap 9, pp 201–230

[80] Zimek A, Vreeken J (2015) The blind men and the elephant: On meeting the problem of multiple truths in data from clustering and pattern mining perspectives. Machine Learning 98(1–2):121–155, https://doi.org/10.1007/s10994-013-5334-y

[81] Zimek A, Assent I, Vreeken J (2014) Frequent pattern mining algorithms for data clustering. In: Aggarwal CC, Han J (eds) Frequent Pattern Mining, Springer, chap 16, pp 403–423

[82] Zou H, Xue L (2018) A selective overview of sparse principal component analysis. Proc IEEE 106(8):1311–1320

Fuzzy C-Means Clustering: Advances and Challenges (Part II)

Janmenjoy Nayak, H. Swapna Rekha, and Bighnaraj Naik

1 Introduction

After the revolutionary development of the concept "fuzzy set" by Prof Lotfi Zadeh in 1965, in 1969 Ruspini gave the initial basis for the developmental progress in fuzzy clustering [1]. Later in 1973, Dunn [2] as well as Bezdek proposed the approach for the fuzzy C-Means model. So, the ultimate basis for the development of FCM is the fuzzy set, and it provided the real framework for distinguishing among points and clusters. However, the variation of FCM called FKM (fuzzy k-means) [3, 4] is a soft clustering method based on the degree of membership. Many of the approaches of FCM have been developed [5] and investigated [6]. Normally, FCM works as a distance-based clustering approach with computation of membership degrees, and the degree of membership decides the membership of data to the cluster group. In fact, for that distance measure also, researchers have used different distance-based methods [7] for solving a variety of applications. With many advantages such as solution for nonlinear, overlapped data, assignment of data point to each of the cluster class, etc., FCM supersedes over other hard clustering methods like k-means clustering and has always been a first and foremost practical-based method for complex ambiguous data. Moreover, it is praiseworthy to mention that till date more than 30 variants have been developed for FCM

J. Nayak (✉)
Department of Computer Science, Maharaja Sriram Chandra Bhanja Deo (MSCB) University, Baripada, Odisha, India

H. S. Rekha
Department of Information Technology, Aditya Institute of Technology and Management (AITAM), Tekkali, K Kotturu, India

B. Naik
Department of Computer Application, Veer Surendra Sai University of Technology, Burla, India

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_12

algorithm. And, so far it has been one of the most successfully used clustering algorithms in many diversified applications across all disciplines of engineering domain. Most of the significant contributions of FCM in between the years 2000 and 2014 are highlighted by Nayak et al. [8]. However, some of the limitations such as initial cluster selection, computing the optimal clusters, receptive to noisy data, etc. of traditional FCM algorithm have always been an open research issue to develop a new method or its variant by fine tuning of the controllable parameters. Moreover, FCM suffers from major limitations like the inability in dealing with a high-dimensional dataset. FCM is based on membership functions, and the factors like the number of prototypes and the number of dimensions of high-dimensional data have a high impact on the membership function. Also, the impact of the number of dimensions highly affects the objective function of FCM. Some of the methods like FCM integrated with polynomial fuzzifier function (PFF-FCM) and PFCM integrated with a noise cluster (PNFCM) [9] and FCM with simulated annealing [10] are proposed to deal with such issues of FCM.

Being a soft clustering algorithm, FCM has remained on the hot seat for clustering different complex data. Over 6,37,000 articles (searched with keyword: fuzzy C-Means clustering algorithm) have been published till 2020. This clearly signifies the popularity of FCM in solving many real-life applications, and it is noteworthy to mention that FCM is one of the most successful clustering algorithms than others. Taking into this account, this chapter illustrates about some of the recent developments using FCM in last five years.

The major objective of this chapter is to provide a review of FCM, its development, and challenges in the recent years. First, we describe the structure of the conventional FCM along with its implementation. Then, the analysis is expanded by describing about the advancement and variants of conventional FCM along with its application areas. Furthermore, a systematic analysis of the utilization of FCM in addressing the challenges of various application areas has been presented. Moreover, a critical analysis on the number of articles published in the FCM, its variants of FCM, and application areas has been depicted to show the growth of FCM. Finally, this chapter serves as basis of information for the technocrats and researchers to apply FCM and its variants to provide novel solution over the conventional methods in various application areas.

The remaining sections are segmented as follows: Sect. 2 depicts about the working of original FCM along with its variations/advancements. Section 3 elaborates the application areas such as Image Analysis, Intrusion Detection System, Clustering and Classification, and Image Analysis. Section 4 discusses the critical analysis of the FCM and its variants. Finally, Sect. 5 describes the conclusion with some future challenges.

2 Structure of Classical FCM

Initially, FCM algorithm has been suggested by Dunn in 1973, and then it has been modified in 1981 by Bezdek. FCM is a leading momentous clustering technique that

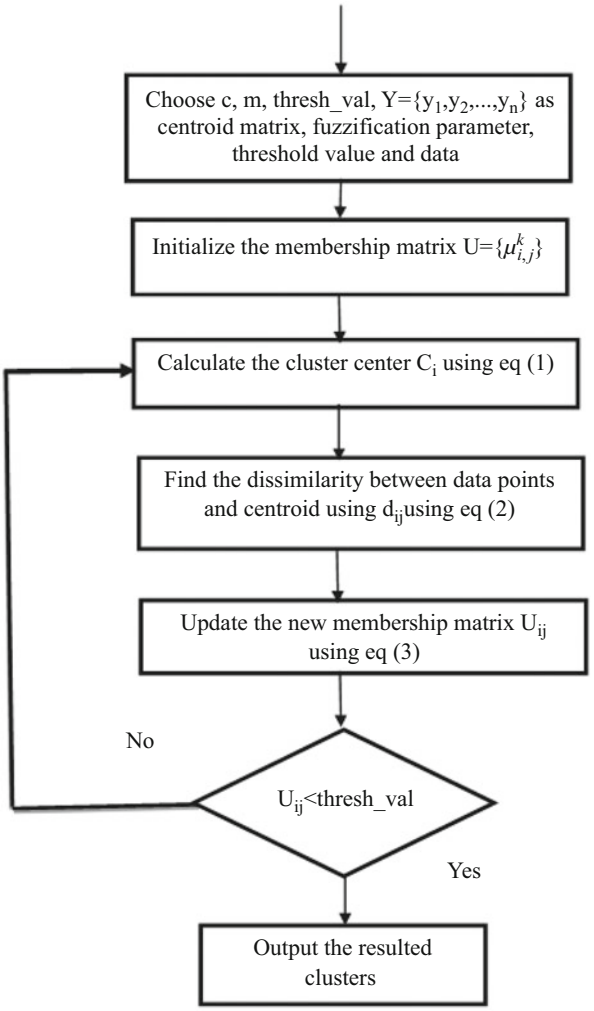


Fig. 1 Flowchart of the FCM algorithm

makes use of fuzzy membership for assigning the degree of membership to each class [11]. The process of forming new clusters from the data points having close membership values is considered as the major advantage of FCM algorithm [12, 13]. The fuzzy membership function, partition matrix, and the objective function are considered as the essential operators in FCM. The detailed steps for FCM algorithm have been depicted in [8]. The following flowchart in Fig. 1 demonstrates the functioning of the basic conventional FCM algorithm.

$$C_i = \frac{\sum_{j=1}^{n} U_{ij}^m \, x_j}{\sum_{j=1}^{n} U_{ij}^m} \tag{1}$$

$$d_{ij} = \| \, C - X \, \| \tag{2}$$

$$U_{ij} = 1/\sum_{k=1}^{c} [d_{ij}/d_{kj}]^{2/(m-1)} \tag{3}$$

C_i is the p-dimension center of the cluster i, d_{ij} is the distance between object x_j and cluster center C_i, and u_{ij} is the degree of membership of x_j in ith cluster.

2.1 Variants and Advancements of FCM

Researchers have also proposed different distinct interpretations of FCM with good performance and are utilized in a varied number of application domains to resolve different problems in clustering. All these distinct types of variants have been successfully utilized for solving a few differences and to obtain proper outcomes accordingly. This section discusses in detail about some of the variants of FCM on which major research work has been carried along with its pros over the traditional FCM. Figure 2 depicts the different variants of FCM.

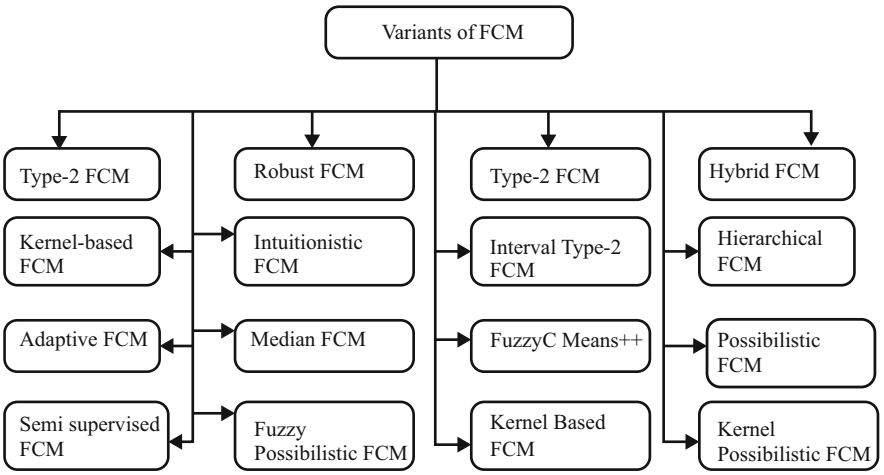


Fig. 2 Variants of FCM

2.1.1 Intuitionistic FCM (IFCM)

Based on the intuitionistic fuzzy set theory, IFCM is considered as the variant of traditional FCM. The conventional FCM has failed to handle the hesitation in medical images because of the existence of vagueness in gray levels, object boundary, and so on. To trounce this drawback, a direct method based on IFCM is proposed by Wang et al. [14] to handle the uncertainty and ambiguity associated with real data. This approach uses an intuitionistic fuzzy triangular product and square product to cluster the input data. The seeds are altered and approximation of membership degree for each intuitionistic fuzzy set (IFS) is performed at each stage of the IFCM. Finally, the IFSs are clustered according to the approximated membership degree. In IFCM [15], membership matrix is defined as

$$u_{ij}\ (k) = 1/\sum_{r=1}^{c}\left(\frac{d_1\left(z_j v_j\ (k)\right)}{d_1\left(z_j v_r\ (k)\right)}\right)^{\frac{2}{(m-1)}} \tag{4}$$

where $d_1\left[Z_j, v_r(k)\right]$ is the distance between the sample $Z_j$and the cluster centroid v_i and u_{ij} is the membership degree of the jth sample Z_j to the ith clustering seed v_i.

For medical image segmentation, a modified intuitionistic fuzzy C-Means clustering (IFCM) algorithm is proposed by Kumar et al. [16]. To compute non-membership value, the proposed approach makes use of Sugeno's and Yager's IFS generators. To avoid the dependency on the fuzzy membership function, a novel clustering algorithm called generalized rough intuitionistic fuzzy C-Means (GRIFCM) has been introduced by Namburu et al. [17] for brain magnetic resonance (MR) image segmentation. Furthermore, the algorithm has been assessed using simulation, and the experimental results show the superiority of GRIFCM over existing k-means (KM), FCM, rough fuzzy C-Means (RFCM), generalized rough fuzzy C-Means (GRFCM), soft–rough fuzzy C-Means (SRFCM), and rough intuitionistic fuzzy C-Means (RIFCM) algorithms. Balasubramaniam et al. [18] have proposed an intuitionistic fuzzy C-Means color clustering algorithm for segmenting the nutrient deficiencies in crop images. Moreover, from the simulation results, it is evident that the proposed algorithm provides better results when compared with existing k-means, fuzzy k-means, PCA, regularized expectation maximization, and FCM algorithms.

2.1.2 Fuzzy Possibilistic C-Means

In the clustering problem as memberships and typicalities play a vital role to obtain the correct feature of data substructure, Pal [19] has proposed a fuzzy possibilistic C-Means (FPCM) algorithm that integrates the features of both fuzzy and possibilistic C-Means algorithm to extract the correct features of the data structure. The FPCM approach is used to resolve the coincident clusters problem of PCM as well as noise

sensitivity shortcoming problem of FCM. The objective function of FPCM [21] is given as

$$J_{FPCM}\left(U, T, V\right) = \sum_{i=1}^{c}\sum_{j=1}^{n}\left(u_{ij}^{m} + t^{n}\right) d^{2}\left(x_{j}, v_{i}\right) \tag{5}$$

where V is the prototype of clusters, U is the fuzzy partition matrix, u_{ij} is the degree of membership, and $d^2(x_j, v_i)$ is the distance between object x_j and cluster center v_i.

As medical segmentation has a vital role in image analysis task, Gomathi et al. [20] have proposed a modified Fuzzy Possibilistic C-Means (MFPCM) algorithm to overcome the limitations of conventional FCM and FPCM. The proposed method is developed by changing the distance measurement of the classical FCM algorithm and exhibits resistance to noise effect during image segmentation. To obtain enhanced quality clustering results, Saad et al. [21] have proposed a modified Fuzzy Possibilistic C-Means (MFPCM) algorithm. Furthermore, numerical simulation of the proposed method displays accurate clustering results when compared to traditional FCM and FPCM. A novel modified Fuzzy Possibilistic C-Means (MFPCM) clustering algorithm has been developed by Ganesan et al. [22] to obtain color image segmentation of noisy color images. Furthermore, the experimental results demonstrate that the projected method exhibits robustness to noise and executes at a faster rate compared to conventional approaches. Praneesh et al. [23] have proposed a modified Fuzzy Possibilistic C-Means (MFPCM) clustering algorithm to extract the text by segmenting color-based comic images. Providing reliable results and obtaining non-overlapping images are considered as the major objectives of the proposed method. Furthermore, analytical results reveal that the proposed method enhances image precision and reduces computation time.

2.1.3 Kernel-Based FCM

In recent years, for analyzing the high-dimensional medical databases, mathematical algorithm-based diagnosing system has been proposed by the researchers. Fuzzy clustering techniques are not potential for high-dimensional databases having more similar objects. As kernel functions have the potential of accessing the available information from high-dimensional databases [24, 25], kernel-based FCM algorithms have been developed as a variant of FCM. As conventional FCM uses squared-norm to determine the affinity among prototypes and data points, it can be adequately used in clustering spherical clusters only. Therefore, it becomes difficult for the original FCM to perform nonlinear mapping to high-dimensional space. This can be achieved through KFCM by using kernel-induced metric in place of Euclidean norm metric in FCM without expanding the number of parameters. The objective function of KFCM [74] is depicted as

$$J_{KFCM} = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{m} \parallel \emptyset(x_j) - \emptyset(v_i) \parallel^2 \tag{6}$$

where $\parallel \emptyset(x_j) - \emptyset(v_i) \parallel^2 = K\left(x_j, x_j\right) + K\left(v_i, v_i\right) - 2K\left(x_j, v_i\right)$

For the analysis of high-dimensional medical databases, Kannan et al. [26] have proposed a fuzzy clustering algorithm by integrating Laplacian kernel-induced distance, Canberra distance, possibilistic memberships, and fuzzy memberships. The proposed method has been evaluated on a high-dimensional breast cancer database, and the experimental results reveal the performance of proposed methods through clustering accuracy. For image clustering, a novel approach namely kernel fuzzy C-Means clustering (GKFCM) algorithm has been developed by Kalam et al. [27]. The purpose of the anticipated method is to deliver enhanced segmentation results for images impaired by noise. To optimize the initial clustering center and to classify the data, a novel kernel-based fuzzy C-Means clustering based on fruit fly (FOAKFCM) algorithm has been proposed by Wang et al. [28]. The proposed method reduces the drawbacks of FCM and enhances clustering efficiency.

2.1.4 Type-2 FCM (T2FCM)

With the original FCM, undesirable clustering results may be produced while considering noisy input data. In current years, type-2 fuzzy logic has been widely applied in many engineering domains due to its robustness. Due to the rapid progression in pattern recognition techniques, clustering has been extensively utilized on various disciplines including general type-2 fuzzy sets which are capable of handling the noise and uncertainty. As clusters obtained by T2FCM tend to converge to more desirable locations than type-1, it is considered as an improved extension of FCM. Torshizi et al. [29] have proposed a powerful and specific similarity measure among a general type-2 fuzzy set. In general type-2 FCM, an optimal number of clusters can be obtained using the general type-2 fuzzy cluster validity index. Furthermore, numerical comparisons show the aspect and robustness of the proposed approach. The objective function of T2FCM [30] is represented as

$$J_{T2FCM} = \sum_{i=1}^{c} \sum_{j=1}^{n} a_{ij}^{m} \parallel x_j - v_i \parallel^2 \tag{7}$$

where $a_{ij}^{m} \parallel x_j - v_i \parallel^2$ represents the type-2 membership function between object x_j and cluster center v_i.

To overcome the uncertainties of medical images, Begum et al. [30] have developed a rough fuzzy C-Means (RFCM) clustering algorithm that incorporates the features of both fuzzy set and rough set. Rough type-2 fuzzy C-Means (RT2FCM) method is an extension of the generalized RFCM algorithm with the type-2 membership function. In contrast to FCM, T2FCM, and RFCM, RT2FCM

performs superior detection of abnormal tissues. Linda et al. [31] has developed a novel approach namely the general type-2 fuzzy C-Means (GT2FCM) algorithm for finding uncertainty in fuzzy clustering. The GT2FCM algorithm has proven to be a robust algorithm in situations where noisy and inadequate data training is available.

2.1.5 Interval Type-2 FCM (IT2-FCM)

Since FCM clustering techniques cannot be used for estimating the number of clusters automatically, Interval type-2 FCM (IT2-FCM) has been developed by Hwang et al. [32]. IT2-FCM utilizes two fuzzifier coefficients and computes two membership values for each pattern for handling the uncertainty in various applications such as image segmentation and land cover classification problems. To create FOU, IT2-FCM [33] makes use of two fuzziness parameters m1 and m2. The utilization of two fuzziness parameters generates two distinct objective functions, as represented in Eqs. 8 and 9.

$$J_{m1}\,(U, v) = \sum_{k=1}^{n}\sum_{i=1}^{c} (u_{ik})^{m_1}\, d_{ik}^2 \tag{8}$$

$$J_{m2}\,(U, v) = \sum_{k=1}^{n}\sum_{i=1}^{c} (u_{ik})^{m_2}\, d_{ik}^2 \tag{9}$$

where d_{ik} is the distance between the pattern x_k and the centroid v_i, C is the number of clusters, and n is the number of patterns.

To handle problems of land cover classification from multi-spectral satellite images, interval type-2 fuzzy C-Means clustering (IT2-FCM) that makes use of spatial information has been refined. To cope up with ambiguity of real data, the idea of collaborative clustering using interval type-2 fuzzy C-Means clustering (IT2-FCM) algorithm has been recommended by Dang et al. [33]. In medical classification, to handle high-dimensional data challenge and ambiguity, an automated classification method using wavelet transformation (WT) and interval type-2 fuzzy logic system (IT2FLS) has been proposed by Nguyen et al. [34]. The proposed method makes use of hybrid learning process that combines both the features of unsupervised structure learning by the FCM algorithm and supervised parameter adapted by genetic algorithm.

2.1.6 Hierarchical FCM (HFCM)

The low-contrast images and noise cannot be segmented properly by FCM, as it uses nonrobust Euclidean distance as dissimilarity function. For the improvement in accuracy and robustness of distance, a generalized hierarchical fuzzy C-Means

(GHFCM) algorithm that makes use of sub-FCM distance function has been recommended by Zheng et al. [35]. In HFCM, the effect of low contrast can be reduced by hierarchical strategy, and noise effect can be reduced by means of neighborhood pixel's information.

Generally, the HFCM is implemented in two levels. In the first level, data are produced by J clusters. While in the second level, data are produced by class-labelled sources within each cluster j which in turn form the sub-clusters of the large cluster. The HFCM [35] objective function is given as

$$J_{mn} = \sum_{i=1}^{N} \sum_{j=1}^{J} \sum_{k=1}^{K} u_{ij}^{m} v_{ijk}^{n} \bar{d}_{ijk} \tag{10}$$

where $\bar{d}_{ijk}$ represents the sub-distance function, and v_{ijk}^{n} represents sub-membership.

To obtain the accurate segmentation of magnetic resonance (MR) images, an improved anisotropic multivariate student t-distribution-based hierarchical fuzzy C-Means (IAMTHFCM) method has been suggested by Chen et al. [36]. The limitations of conventional FCM such as less robustness to outliers, weak edges, noise, and less accuracy can be overcome using the IAMTHFCM algorithm. A novel Hierarchical Hyperspherical Divisive Fuzzy C-Means (H2D-FCM) has been developed by Bordogna et al. [37]. The automatic finding of the number of clusters, fuzzy partition, and conditional splitting at each node is considered as the main features of the proposed algorithm. To overcome the drawbacks of over-smoothness in image segmentation, a novel non-local-based spatially constrained hierarchical fuzzy C-Means NLSCHFCM algorithm has been suggested by Chen et al. [38]. The limitations of existing methods such as the effect of noise, data preserving, and inaccuracy can be overcome using the NLSCHFCM algorithm. The robustness and accuracy can be enhanced by using a hierarchical strategy that makes use of improved distance function itself as a sub-FCM.

2.1.7 Robust FCM

From the past few decades, several robust fuzzy clustering algorithms are developed by the researchers to partition data distressed by noise as well as outliers. Among them, the widely used robust clustering algorithm namely a robust version of FCM (robust FCM) has been proposed by Dave [39]. The objective function of robust FCM [41] is given as

$$J\left(V, U, X\right) = \sum_{i=1}^{C} \sum_{j=1}^{M} u_{ij}^{m} \bullet d^{2}\left(v_{i}, x_{j}\right) + \sum_{j=1}^{M} u_{*j}^{m} \bullet \delta^{2} \tag{11}$$

where M represents the number of objects, C represents the number of classes, m represents fuzzification parameter, u_{ij} represents membership degree, and $d(v_i, x_j)$ represents the distance between cluster v_i and object x_j.

To enable the effective segmentation of SAR images, Wan et al. [40] have proposed a novel robust FCM algorithm based on Bayesian nonlocal spatial information (RFCMBNL). Furthermore, experimental results reveal that the projected algorithm simultaneously performs effective segmentation of SAR images by assuring noise resistance and edge detail preservation of the image. Robust fuzzy C-Means (robust FCM) algorithm has been recommended by Cimino et al. [41] for the automatic determination of the most feasible δ for the specific application. The method has been evaluated on three datasets, and the results reveal that it can obtain optimal performance in minimum computational time. For enhancing the quality of image segmentation and to minimize the effect of noise, a vital fast and robust FRFCM algorithm has been introduced by Lei et al. [42]. Furthermore, the developed method exploits local spatial constraints by applying membership filtering.

2.1.8 Semi-supervised FCM

In a heterogeneous environment, the existing clustering techniques have proven to be difficult in determining the scattered regions of an image. Because of the selection of different features, the conventional FCM is not considered as the better option for the identification of accurate clustering. Therefore, by combining the semi-supervised learning with FCM, the detection accuracy of clustering can be enhanced. The semi-supervised learning algorithm was developed by Pedrycz & Waletzky [43] by including supervised learning as the second term in the unsupervised learning objective function. Therefore, the semi-supervised FCM [43] objective function is defined as

$$J = \sum_{i=1}^{c}\sum_{k=1}^{N} u_{ik}^{m} d_{ik}^{2} + \sum_{i=1}^{c}\sum_{k=1}^{N} (u_{ik} - f_{ik} b_k)^{m} d_{ik}^{2} \tag{12}$$

where f_{ik} = membership value of labelled data pattern k in cluster c_i, b_k = Boolean vector, c = the number of clusters, N = the number of data patterns u_{ik} = membership value of a data pattern k, and m = fuzzifier parameter

For the automatic classification of the Nottingham Tenovus Breast Cancer dataset, a semi-supervised Fuzzy C-Means (ssFCM) algorithm is explored by Lai et al. [44]. In the field of social signal processing, human emotion recognition is considered as an important area of research. Therefore, Liliana et al. [45] have proposed methods namely Active Appearance Model (AAM) and ssFCM for identifying the facial expression in detecting human emotions. Furthermore, experimental results reveal that the proposed methods obtain 80.71% accuracy rate compared to the existing fuzzy inference system. To solve the problem of local optima, semi-supervised learning that combines the features of FCM has been

developed by Guorui et al. [46]. Furthermore, the method has been evaluated on the KDD CUP 99 dataset, and the results reveal that the projected method greatly enhances the operation of intrusion detection

2.1.9 Hybrid FCM

From the past few years, hybrid FCM approaches have been determined as dynamic approaches for the diagnosis of related data in various clusters. Accordingly, several hybrid applications have been suggested by the researchers in FCM by considering benefits and by avoiding the difficulties of different methods. To optimize the energy in wireless sensor networks, Bouyer et al. [47] have proposed the FCM algorithm along with hybrid leach protocol. To balance energy usage of the CHs and for enhancing the lifetime of the network, the FCM algorithm has been used. They also explained that the usage of FCM in wireless sensor networks supports changing the LEACH protocol parameters while execution. Furthermore, experimental results show that the lifetime of a network has been enhanced by the usage of the hybrid algorithm. For clustering heterogeneous data on cloud computing, Li et al. [48] have proposed a privacy-preserving high-order neuro-fuzzy C-Means (PPHOFCM) algorithm. The approach has been used to cluster the heterogeneous dataset. For catching the correlations in the high-order tensor space, they have used the tensor distance. To enhance the clustering capability for enormous heterogeneous data from IoT, cloud computing has been employed. To safeguard the private data while operating the high-order neuro-fuzzy C-Means algorithm on cloud computing, the BGV encryption scheme has been employed. Furthermore, the efficiency of PPHOFCM has been verified by conducting experiments on two real IoT datasets. To enhance the accuracy of the detection system, Pandeeswari and Ganesh Kumar [49] have proposed a hybrid algorithm that makes uses of the FCM and Artificial Neural Network (FCM-ANN). They have also compared the proposed method with the Naïve Bayes classifier and Classic ANN algorithm. Moreover, experiments have been conducted using DARPA's KDD cup dataset 1999. Furthermore, the results reveal that the proposed method is having high detection accuracy and low false alarm rate when compared with the Naïve Bayes classifier and Classic ANN. Lotfan et al. [50] have proposed enhanced fuzzy C-Means based on the heuristic subtractive approach (FCM-S) for diagnosing bearing faults. They have also explained that the performance and convergence of clustering can be highly enhanced by the application of hybrid C-Means-Subtractive (FCM-S). The method provides better diagnosing of bearing faults when compared with conventional FCM. A novel approach of fuzzy rough feature selection has been proposed by Zhao et al. [51]. To enhance the classification accuracy and to decrease the data dimension, this approach associates the membership function decision approach of fuzzy C-Means clustering and fuzzy equivalence to the original selection. Moreover, the proposed approach is able to provide smaller subsets and high classification accuracies (more than 1% average can be achieved). Table 1 displays the analysis of other literature reviews on variants of FCM.

Table 1 Some other literature reviews on variants of FCM

Author and year	Algorithm used	Compared method	Application area	Results	Ref
Karimi & Sensoy 2020	FCM-PSO (Particle Swarm Optimization)	Full Monte Carlo method	seismic fragility curves	Mean=0.9977, Standard Deviation=0.4575	[52]
Chu et al. 2020	LASSO-FCM-DBN (Deep Belief Network)	ANN (Artificial Neural Network), DBN, LASSO-FCM-ANN	Streamflow prediction	Calibration period MAE =0.271, RMSE=330, R2=0.894, Validation period MAE=0.283, RMSE=305, R2=0.860	[53]
Vandarkuzhali et al. 2020	OBMFCM (Optimization-based Modified FCM) and SVM (Support Vector Machine)	OMBFCM, FCM	Detection of fovea region in retinal images	High accuracy in classifying fovea region in retinal images	[54]
Majdi et al. 2019	FCM optimized by Evolutionary algorithms	FCM without optimization	Predict the deformation modulus of rock masses	RMSE: FCM optimized by GA (Genetic Algorithm) =3.16, FCM optimized by PSO=3.01	[55]
Anter et al. 2019	Fuzzy C-Means and adaptive watershed algorithm	Adaptive Threshold, Region growing	CT liver tumor Segmentation	Dice coefficient = 92.88%	[56]
Hu et al. 2019	FCM-MMEMD	EMD (Empirical Mode Decomposition), EEMD (Ensemble EMD), MEMD (Modified EMD), MMEMD (Multi-masking EMD)	Wind turbine bearing fault diagnosis	RMSE1=0.1880, RMSE2=0.2138, RMSE3=0.1152	[57]

Kumar et al. 2019	MIFCM (Modified Intuitionistic FCM)	FCM, IFCM, PIFCM (Possibilistic FCM)	Image Segmentation	Dice Score of White matter=0.9267	[58]
Zhang et al. 2019	BBO-FCM (Biogeography-based Optimization-FCM)	FCM, AFSA-FCM (Adaptive Artificial Fish Swarm-FCM), ABC-FCM (Artificial Bee Colony-FCM), PSO-FCM	Image Segmentation	Classification entropy on image Lena=8.5050 e−005	[59]
Zeraatpisheh et al. 2019	DSMART-FCM-KM (Disaggregation and Harmonization of Soil Map Units Through Resampled Classification Trees-FCM-k-means)	FCM, KM	Disaggregating and updating a legacy soil map	Disaggregated soil maps obtained overall accuracy of 36% and 33% at great group and subgroup levels	[60]
Azzaoui et al. 2019	Wavelet Transform & FCM	FCM	Fault diagnosis	Classification rate=74.17%, Sensitivity=100%	[61]
Wu et al. 2019	FKMFCM-MCC	KM, FKM (Fuzzy KM), AFKM (Agglomerative FKM), SFKM (Sparse FKM), RSFKM (Robust and Sparse FKM)	Cluster fuzziness	Accuracy of Coll20=67.23±2.05	[62]
Rezaee et al. 2018	FCM and data envelopment analysis (DEA) and ANN	FCM	Online prediction performance of companies in stock exchange	Selects best cluster	[63]

(continued)

Table 1 (continued)

Author and year	Algorithm used	Compared method	Application area	Results	Ref
Nayak et al. 2018	ETLBO-FCM (Elicit Teaching Learning-Based Optimization-FCM)	FCM, GA-FCM, PSO-FCM, IPSO-FCM (Improved PSO-FCM), TLBO-FCM	Data clustering	Xie Beni (XB) index of iris data=0.0280	[64]
Alsmirat et al. 2017	IT2FCM using hybrid CPU-GPU	FCM, T2FCM, PFCM, IT2MFPCM	Medical image segmentation	Performance enhances by 8.9x	[65]
Nayak et al. 2017	CRO-FCM (Chemical Reaction Optimization-FCM)	GA-k-means, PSO-k-means, ETLBO-k-means, TLBO-k-means, GA-FCM, PSO-FCM	Cluster analysis	Intra cluster distance=462.421, Inter cluster distance=750.732	[66]
Ouma et al. 2017	Fuzzy C-Means clustering and morphological reconstruction	FCM	Pothole detection	Average Dice Coefficient=87.5%	[67]
Fatehi et al. 2017	ssFCM (Semi Supervised FCM)	FCM	Classification	VXB=39.6, VPC=0.67	[68]
Kumar et al. 2017	IABCFCM (Improved Artificial Bee Colony FCM)	FCM, ABC	Clustering	Low standard deviation=0.0170, Low error rate=16.6378	[69]
Rubio et al. 2017	IT2FPCM	IT2FCM	Clustering	Better clustering performance	[70]
Mai et al. 2015	IIT2-FCM (Information IT2-FCM)	k-means, FCM, IT2-FCM	Land cover classification	Mean=9.48, Standard deviation=4.14	[79]
Kisi et al. 2016	ANFIS-FCM (Adaptive Neuro-Fuzzy Embedded-FCM)	ANFIS-GP (Grid Partition), MLP (Multi-layer Perceptron), SRC (Sediment Rating Curve)	Assessment of the suspended sediment	Station 06088300, RMSE=0.806 Station 06088500, RMSE=0.716	[71]

Bai et al. 2016	FCM and rough set	–	Complex investment decisions	Better decisions in environmental and business perspective	[72]
Chattaraj et al. 2016	Kernel-based FCM	FCM	Segmentation of mammographic images	Better segmentation results	[73]
Ding et al. 2016	GAKFCM (Kernel-based fuzzy C-Means clustering algorithm based on genetic algorithm)	FCM, KFCM	Clustering	Precision of Health dataset=98%	[74]
Fathabadi et al. 2016	dFCM+ANN (Dynamic FCM+ANN)	Switching algorithm	Power distribution network reconfiguration	Average processing time=0.51s	[75]
Ahmmed et al. 2016	TKFCM (Template-based k-means and FCM)	K-Mean, FCM, Thresh-holding, Region Growing, second order+ANN, Texture combined+ANN	Tumor detection	Accuracy=97.1%, Specificity=100%, Sensitivity=96.67%	[76]
Keskin 2015	Fuzzy DEMATEL and fuzzy C-Means algorithm	–	supplier evaluation and selection	Used as decision-making tool for supplier selection decisions.	[77]
Abdel-Maksoud et al. 2015	KIFCM (k-means integrated with FCM)	KM, EM (Expectation Maximization), MS (Mean Shift), FCM	Brain tumor segmentation	Accuracy on dataset1=90.5%, dataset2 & 3=100%	[78]
Tang et al. 2015	FCM-Genetic Algorithm	FCM	Missing traffic volume data estimation	Correlation coefficient range from 0.9703 to 0.9879	[80]

(continued)

Table 1 (continued)

Author and year	Algorithm used	Compared method	Application area	Results	Ref
Gupta et al. 2015	GKFCM_S1 (Gaussian Kernel FCM_S1) GKFCM_S2	FCM_S1, FCM_S2	Image Segmentation	Accuracy of GKFCM_S2=98.1947, GKFCM_S1=97.4654	[81]
Singh et al. 2015	FCM-SVM	–	Detection of brain tumor in MRI images	Accuracy of linear Kernel function=91.66%	[82]
Izakian et al. 2015	FCM–FCMdd (FCM-Fuzzy C-Medoids)	FCM, FCM-EU (FCM-Euclidean distance), FCMdd	Clustering of time series data	Generates superior clusters	[83]
Silva Filho et al. 2015	FCM-IDPSO (FCM-Improved Self Adaptive PSO, FCM2-IDPSO	HPSOFCM Hybrid PSOFCM), CPSFC (chaotic particle swarm fuzzy clustering)	Fuzzy clustering	Standard deviation of dataset1=0.0017, dataset2=0.0021	[84]
Mekhmoukh et al. 2015	Improved FCM-based PSO	FCM, FCM_S1, RFCM (Robust FCM), KPCM (Kernel possibilistic C-Means)	Image Segmentation	Dice Index of white matter=0.9771 and Gray matter=0.9761	[85]
Gupta et al. 2015	FCM-GENSM (FCM-Genetic-based similarity measure)	GENSM	Optimal similarity measures and similarity metrics with the movie lens dataset	for 30 iterations MAE=0.53, Precision=0.81	[86]
Stetco et al. 2015	Fuzzy C-Means++	FCM	Clustering	Wine dataset & Iris dataset: Xie-Beni=0.0009127211, Spam dataset: Xie-Beni=1.084566e-05	[87]
Son et al. 2015	DPFCM (Distributed Picture FCM)	CDFCM (Color Differentiated FCM)	Clustering Quality	Better Clustering quality	[88]

(where RMSE: Root Mean Square Error, MAE: Mean Absolute Error, PC: Partition Coefficient)

3 Applications of FCM and Its Variants

This section gives an overview on the literature survey of various papers published on FCM from 2015 to 2020. To solve the real-world problems, FCM has been lucratively applied in various fields of application such as computer science, mathematics, and other areas of engineering, etc. FCM and its variants have been extensively applied in the application domain such as neural networks, clustering and classification, Image analysis, fault diagnosis, intrusion detection system, etc.

3.1 FCM and Neural Networks

In power quality (PQ) classification to identify the important training data, a novel data selection approach has been developed by Manimala et al. [89]. This approach makes use of FCM along with two probabilistic neural networks and support vector machines. Minimizing the computation complexity and execution time and increasing the existing PQ classification system accuracy are considered as the main objectives of the proposed method. To solve the problem of incomplete data, Zhang et al. [90] have developed a robust method that makes use of FCM along with probabilistic information granules based on the nearest neighbors of incomplete data. Furthermore, the method uses Lagrange multipliers to optimize the clustering model. An automated approach for retinal blood vessel segmentation has been proposed by Hassanien et al. [91]. In order to find the coarse vessels, the proposed method integrated artificial bee colony optimization with FCM in the first level. To enhance the segmentation results, pattern search has been applied in the second level. Table 2 displays the other literature survey on FCM and neural networks.

3.2 Image Analysis Using FCM

To resolve the problem of delineating various tissues from brain image, a novel method incorporating the features of bias-field corrected fuzzy C-Means and level set segmentation has been recommended by Agarwal et al. [96]. This approach produces better results when compared to individual FCM and level set segmentation. To effectively segment the brain magnetic resonance images in clinical diagnosis, Vermal et al. [97] have proposed an improved intuitionistic fuzzy C-Means (IIFCM) clustering algorithm. The objective of the proposed method is to provide noise resistance and certainty on the boundary between distinct tissues of brain images.

Table 2 Some other literature reviews of FCM and neural networks

Author and Year	Algorithm used	Application area	Results	Ref
Mansouri et al. 2018	ANFIS-FCM	Evaluation of peak and residual conditions	Validation phase: RMSE=0.0053, AAE=26.2	[92]
Mahela et al. 2017	S-Transform-Based Ruled Decision Tree and FCM	Recognition of Power Quality Disturbances	Efficiency without noise=99.60 %, With noise=99.30%	[93]
Rajaby et al. 2016	FCM with weighted hue and intensity	Image Segmentation	Reduces number of iterations and avoids wrong centroids	[94]
John et al. 2015	AFCM-CNN	Pedestrian Detection in Thermal Images	Average log-average miss rate =34%	[95]

(where AAE: Absolute Average Error)

Hien et al. [98] have proposed an approach that consists of three stages for resolving edge detection issues in brain images. Initially, the quality of the image has been enhanced using Semi-Translation Invariant Contourlet Transform (STICT). Then, FCM has been applied for segmenting the image. Finally, edges are detected using edge detection method. In Table 3, a list of modified FCM algorithms that have been used in the Image analysis has been depicted.

3.3 Clustering and Classification Using FCM

Li et al. [110] have recommended a new method for resolving the problem of magnetization inversion using the FCM clustering method. Even in the existence of remnant magnetization, the magnetic irregularity can be completely utilized through 3D magnetization. A dynamic method that performs classification through the application of the FCM algorithm to MFCC and LPC features using EmoDB has been recommended by Demircan et al. [111]. Furthermore, experimental results show that the proposed method can achieve a maximum classification success rate through kNN and SVM classifiers. In order to perform the accurate sentiment classification in a parallel network environment, Phu et al. [112] have proposed an FCM method along with Hadoop MAP (M)/REDUCE (R). To find the best number of clusters, an automatic robust learning FCM (RL-FCM) algorithm has been suggested by Yang et al. [113]. Robustness to parameter selection, initialization, and automatic finding of the number of clusters are considered as the main characteristics of the RL-FCM algorithm. Other applications that have been using FCM in clustering and classification have been listed in Table 4.

Table 3 Some other literature reviews of FCM and neural networks

Author and year	Algorithm used	Compared method	Results	Ref
Syed et al. 2019	Spatial Kernel Fuzzy C-Means	–	Speed of divergence and run time is less	[99]
Hu et al. 2019	AKFCM (Adaptive kernel-based FCM)	FCM, KFCM, GKFCM (Gaussian KFCM), EKFCM (Entropy KFCM)	Robust to image segmentation	[100]
Bilenia et al. 2019	Modified FCM	FCM	Better Segmentation results	[101]
Nida et al. 2019	Deep region-based CNN (Convolutional Neural Network) and FCM	ExB, CUMED, Mahmudur, SFU-mial, TMUteam, UiT-Seg, IHPC-CS, UNIST, Jose Luis, Marco Romelli	Accuracy=0.942	[102]
Alsmadi et al. 2018	NFCM (Novel FCM)	FCM, ABCFCM, FAFCM (Firefly FCM)	Similarity index=0.9471, Specificity=0.9412, Sensitivity=0.9592	[103]
Noh et al. 2017	FCM	Sobel detector segmentation	Recall=0.8, Precision=0.9	[104]
Shedthi et al. 2017	FCM	k-means algorithm	Higher clustering performance	[105]
Shang et al. 2016	CKS_FCM (Clone Kernel Spatial FCM)	CS_FCM (Clone Spatial_FCM), CK_FCM (Clone Kernel_FCM), KS_FCM (Kernel Spatial_FCM)	Segmentation accuracy at speckle look1=92.70±1.51, Speckle look 2=97.26±1.76, Speckle look 4=98.85±1.14, Speckle look 6 =99.04±0.46	[106]
Marrone et al. 2016	2DFCM (2-Dimensional FCM)	Pixel-based, Atlas-based, and Geometric-based approach	Accuracy=97.86% Specificity=95.83%, Sensitivity=98.44%	[107]
Feng et al. 2016	NLMD+BCEFCM (Non-local means Denoising + Bias Correction Embedded FCM)	FCM, BCFCM (Bias-Corrected FCM), MICO (Multiplicative Intrinsic Component Optimization), BCEFCM	Run time of Rice like object=0.2003, Geometrical shape=0.0484, X-ray image of vessels=0.0364, MRI image of brain=0.9110	[108]
Adhikari et al. 2015	csFCM (Conditional Spatial FCM)	k-means, FCM, FGFCM (Fast Generalized FCM), sFCM (Spatial FCM), ASIFCM (Adaptive Spatial Information FCM),	Area Under Curve for Cerebro Spinal Fluid=0.9832, Gray Matter=0.9620, White Matter=0.9776	[109]

Table 4 Some other literature reviews of Clustering and classification using FCM

Author and year	Algorithm used	Results	Ref
Jiekang et al. 2020	FCM	For the period of 21:00–22:00, Load loss rate=0.0873, Power loss rate=0.0303	[114]
Ranjbarzadeh et al. 2020	FCM	For Liver Segmentation: Average surface Distance (ASD)=1.1±0.39 mm and Volume overlap error (VOE)=1.8±0.34% For tumor segmentation ASD=1.5±0.55 mm and VOE=9.8±3.9%	[115]
Silva et al. 2019	bdrFCM (Boundary Data Reduction FCM)	bdrFCM on Poker, PAMAP2, MNIST, CoverType and SKIN dataset at 20% interval is 6.62, 19.28, 2.95,4 and 2.15 times greater than original FCM	[116]
Zeng et al. 2019	Multi-kernel IFCM	Dice coefficient of white matter=0.9925 and Segmentation accuracy=0.9857	[117]
Palani et al. 2019	FCM-ARM-DT-CNN (FCM-Association Rule Mining-Decision Tree-CNN)	MSE=0.159, PSNR=57.02 and accuracy=99.54%	[118]
Sharma et al. 2019	FCM	Performance improvement of NBIS matcher is 1.54% to 50.62% and VeriFinger is 1.66% to 8.95%	[119]
Zhao et al. 2018	FCM-Fuzzy membership	Diagnostic accuracy=96%	[120]
Khan et al. 2018	FCM	Overall accuracy=76%	[121]
Haldar et al. 2017	FCM-M (FCM-Mahalanobis Distance)	Number of iterations reduced to an average of 53%	[122]
Benmouiza et al. 2016	FCM	Better results obtained when sizing in hourly solar radiation scale CA=0.91 and CS=3.2	[123]
Lin et al. 2015	nsiibFCM (noise size-insensitive integrity-based FCM)	Segment Accuracy Gaussian with 15% noise=99.68, Salt and Pepper (15%)=99.40	[124]
Yang et al. 2015	Bias-correcting FCM and Gustafson and Kessel (GK)	Error rate of Banana-shaped dataset= 0%, Iris Dataset=10%, Crab dataset=0.09, Flea beetle data set=0.0135	[125]
Rai et al. 2015	FCM	Accuracy=94.58%	[126]
Kisi et al. 2015	LSSVR (Least Square Support Vector Regression) and ANFIS-FCM	Average RMSE: RMA model reduced by 15.6% Besiri and Baykan stations reduced by 12.4%	[127]
Jagannath et al. 2015	FCM	Identifying retinal images accuracy=94.5%,Classifying normal retinal images accuracy=86%	[128]
Khan et al. 2015	FCM	For YIQ component with number of clusters=3 specificity=93.19%, sensitivity=89.67% and accuracy=92.63%	[129]

3.4 *Intrusion Detection Using FCM*

The development of the Intrusion Detection System is emerging as a prominent research area in the globalization era to protect the data from network attacks. Rustam et al. [130] have made a comparison analysis of IDS using SVM and FCM to obtain better performance and to enhance the accuracy of network attacks. Furthermore, experimental results show that FCM achieves better accuracy compared to SVM. For implementing the Intrusion Detection System, a novel PSO-FCM algorithm has been developed by Zhang et al. [131]. In the proposed method, clustering has been utilized to find intrusion irregularity and classification has been used to find the intruders. The proposed method depicts better performance compared with the k-means and FCM algorithms. Table 5 lists the other analysis performed in intrusion detection using FCM.

Table 5 Some other literature reviews of FCM and neural networks

Author and year	Proposed method	Compared method	Results of proposed method	Ref
Rustam et al. 2020	Intrusion Detection System model with Fuzzy Kernel C-Means	Intrusion Detection System model with Fuzzy Kernel C-Means and Laplacian Score feature selection	Accuracy of all classes (Normal, DoS, Probe, U2R and R2L)=67.06%	[132]
Zhang et al. 2019	FCM-SVM	online sequential extreme learning machine (OS-ELM) and multi-level hybrid (MLH)	Accuracy=99.19%	[133]
Duan et al. 2018	FCM+KNN	FCM, TANN, CANN, FCM+ANN	Accuracy=98.73%, Detect rate=96.23%, False alarm=0.28	[134]
Kumar et al. 2016	FCM and SVM	NB (Naive Bayes)+FCM, SVM+k-means, NB+k-means	Accuracy=99.98%	[135]
Rustam et al. 2015	Fuzzy Kernel C-Means	C4.5 Decision Tree +NB	Detection rate of Normal DoS=100%, Normal-Probe=100%, Normal-U2R=87.64%, Normal-R2L=98.66	[136]
Sampat et al. 2015	Improved Dynamic FCM	Simple k-means, Fuzzy C-Means	True Positive rate of correctly classifying an intrusion=90.45%	[137]

4 Critical Analysis

From the above data, it can be observed that the variants of the FCM can be used to resolve several problems associated with the conventional FCM. Furthermore, the critical analysis provides researchers with the knowledge of the pros and cons of application of FCM and its variants in various application domains. The limitations associated with traditional FCM such as (1) existence of vagueness in gray levels and object boundary, (2) coincident cluster problem, (3) efficient clustering of high-dimensional data, (4) handling of noisy and low-contrast corrupted images, (5) automatic estimation of number of clusters, (6) detection accuracy, (7) highly sensitive to initialization have been successfully resolved using the variants of FCM like Intuitionistic FCM, FPCM, KFCM, T2FCM, IT2-FCM, semi-supervised FCM, etc. Furthermore, hybrid approaches help in enhancing the clustering and optimization capabilities of FCM and its variants by integrating the capabilities of two or more than two approaches.

Moreover, it has been also noticed that enhancing image segmentation by incorporating deep learning modules, performing automatic clustering without having previous information on the number of clusters, enhanced classifier for extracting appropriate features from segmented images, improving strategies for achieving robustness in clustering heterogeneous noisy data in IoT, usage of optimization techniques for improving the performance and automation of clustering algorithms in variants of FCM, usage of 3D evaluation schemes for enhancing the efficiency of MRI brain tumor segmentation, using efficient clustering schemes for increasing the accuracy of feature selection method, application of various data mining techniques to enhance the performance of classifiers, conduction of more studies on adjustment of weight parameters in clustering process for better reflection of similarity patterns can be considered as further scope of research in the implementation of FCM and its variants.

From the above systematic review, it is clear that the applicability of FCM in various applications has been of great perspective. The detailed analytical view of FCM and its variants in various application areas have been interpreted in the further part of the section by considering the papers published in standard journals and conferences such as Springer, Elsevier, ScienceDirect, IEEE Xplore, and so on. The percentage of articles published on variants of FCM in different domains has been depicted in Fig. 3. From the figure, it can be analyzed that a greater number of articles, i.e., 32% of articles, has been published in robust fuzzy C-Means algorithm. Next, the majority of the work has been done using hybrid FCM, i.e., 25%. After hybrid FCM, 22% of the articles have been published using hierarchical FCM. Less than 10% of the articles have been published in other variants of FCM such as IT2FCM, T2FCM, KFCM, FPCM, Intuitionistic FCM, and semi-supervised FCM.

The number of articles published in different application domains of FCM such as Image segmentation, clustering, classification, Intrusion detection system, and neural networks has been represented in Fig. 4. From the figure, it is evident

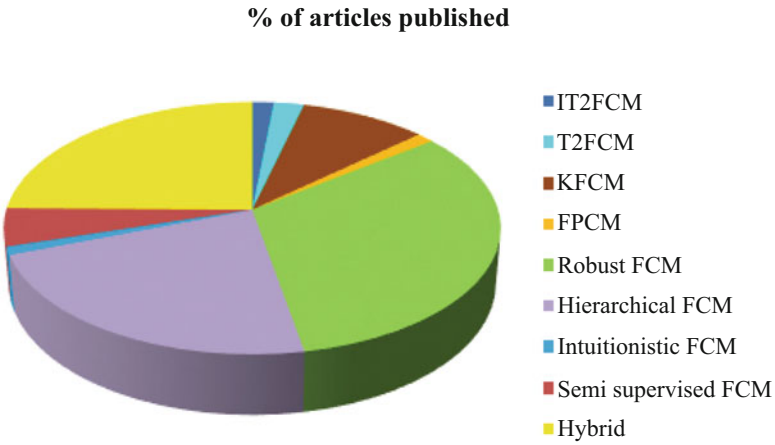


Fig. 3 Usage levels of mostly used variants of FCM

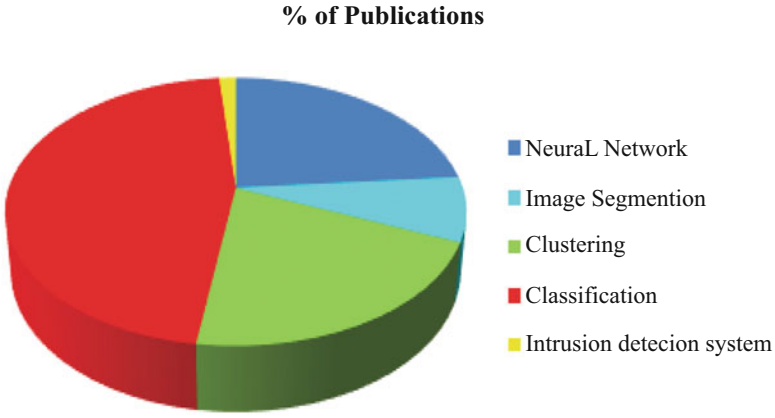


Fig. 4 Articles published in different application areas of FCM

that major research articles, i.e., 46% of the work, have been published in the classification area. After classification, 24% of the work has been published in the neural network domain. Next, the major research articles have been published in the clustering domain, i.e., 21%. Less research articles using FCM have been published in the area of Image segmentation and intrusion detection system.

Finally, Fig. 5 represents the number of articles published using FCM from 2015 to 2020. It is evident from the figure that the growth of FCM has been increasing from 2015 to 2019.

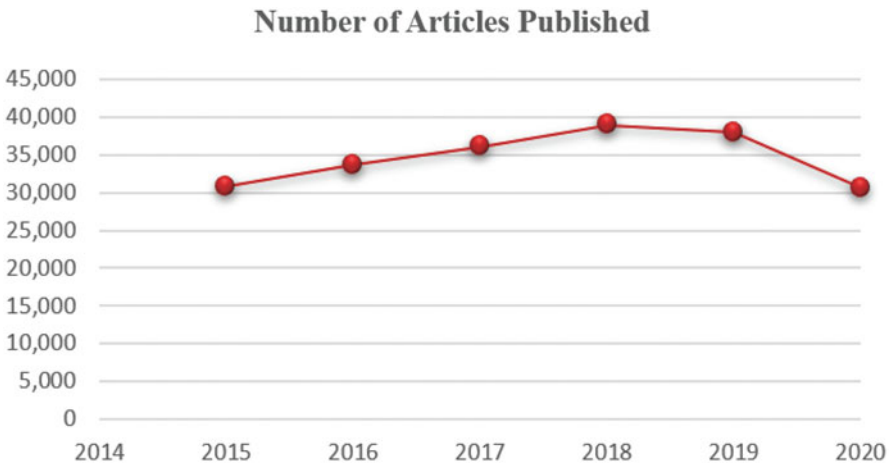


Fig. 5 Statistics of distribution of articles using FCM during 2015 to September 2020

5 Conclusions

From its inception, FCM has been a top prioritized clustering algorithm among all research communities. From the literature study, we are having a clear depiction of the increasing rate of publications related to fuzzy clustering (directly or indirectly). Apart from some of the general problem domains such as classification, clustering, image analysis, fault prediction, mathematical optimizations, forecasting, anomaly detection, medical diagnosis, etc., FCM has been a good choice for big data, swarm intelligence, fuzzy graphs, kernel-based optimization problems, cyber-physical system, robotics, data streaming, etc. in twenty-first century. Table 1 presents the number of variants of FCM been developed for the last five years. Being an objective function-based algorithm, it behaves more naturally as compared to hard clustering. However, some of the limitations like finding the best solution with spherical clusters, deficient to noise sensitivity, more computational time, local minima limit, irregular distribution of membership values to noise data points, etc. have always been a challenging area for focus. Up to some extent, few algorithms like possibilistic C-Means, fuzzy possibilistic C-Means, possibilistic fuzzy C-Means, etc. have remained to solve certain limitations of original FCM. Furthermore, some researchers have fused both the FCM and possibilistic C-Means to develop a hybrid model for better functionality. However, still some of the major issues such as selecting the optimal level of cluster fuzziness, solution for condense data, choosing the membership degree at high noise level, handling more prototypes in large data, etc. needs urgent focus. As future work, it may be applied in some novel application areas such as biology, criminology, cyber-physical system, military applications, instrumental theory, mechanical designs, etc., which may lead as a novel solution over traditional methods.

References

1. Ruspini, Enrique H., James C. Bezdek, and James M. Keller. "Fuzzy clustering: A historical perspective." IEEE Computational Intelligence Magazine 14.1 (2019): 45–55.
2. Dunn, J. C. (1973-01-01). "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters". Journal of Cybernetics. 3 (3): 32–57.
3. J.C. Bezdek, A convergence theorem for the fuzzy isodata clustering algorithms, IEEE Trans. Pattern Anal. Mach. Intell. 2 (1) (1980) 1–8.
4. E.H. Ruspini, A new approach to clustering, Inf. Control 15 (1) (1969) 22–32.
5. X. Cai, F. Nie, H. Huang, Multi-view k-means clustering on big data, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2013.
6. F. Nie, Z. Zeng, W. Ivor, D. Xu, C. Zhang, Spectral embedded clustering: a framework for in-sample and out-of-sample spectral clustering, IEEE Trans. Neural Netw. 22 (11) (2011) 1796–1808.
7. Goel, Sonia, and Meena Tushir. "A new iterative fuzzy clustering approach for incomplete data." Journal of Statistics and Management Systems 23.1 (2020): 91–102.
8. Nayak, Janmenjoy, Bighnaraj Naik, and H. S. Behera. "Fuzzy C-means (FCM) clustering algorithm: a decade review from 2000 to 2014." Computational intelligence in data mining-volume 2. Springer, New Delhi, 2015. 133–149.
9. Winkler R, Klawonn F, Kruse R (2011) Fuzzy C-Means in High Dimensional Spaces. International Journal of Fuzzy System Applications (IJFSA), 1(1), 1–16. https://doi.org/10.4018/IJFSA.2011010101.
10. Liu, Peiyu, et al. "An improved fuzzy c-means clustering algorithm based on simulated annealing." 2013 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). IEEE, 2013.
11. Subhagata Chattopadhyay, Dilip Kumar Pratihar, "A Comparative Study Of Fuzzy C-Means Algorithm And Entropy-Based Fuzzy Clustering Algorithms", Computing and Informatics, Vol. 30, 2011, 701–720.
12. Asyali, M.H., Colak, D., Demirkaya, O., Inan, M.S.: Gene expression profile classification: a review. Curr. Bioinform. 1, 55–73 (2006).
13. Ganesan, P., and V. Rajini. "A method to segment color images based on modified fuzzy-possibilistic-c-means clustering algorithm." Recent Advances in Space Technology Services and Climate Change 2010 (RSTS & CC-2010). IEEE, 2010.
14. Z. Wang, Z. Xu, S. Liu and Z. Yao, Direct clustering analysis based on intuitionistic fuzzy implication, Appl. Soft Comput. 23 (2014), 1–8.
15. Xu, Zeshui, and Junjie Wu. "Intuitionistic fuzzy C-means clustering algorithms." Journal of Systems Engineering and Electronics 21.4 (2010): 580–590.
16. Kumar, SV Aruna, and B. S. Harish. "A modified intuitionistic fuzzy clustering algorithm for medical image segmentation." Journal of Intelligent Systems 27.4 (2018): 593–607.
17. Namburu, Anupama, Srinivas Kumar Samayamantula, and Srinivasa Reddy Edara. "Generalised rough intuitionistic fuzzy c-means for magnetic resonance brain image segmentation." IET Image Processing 11.9 (2017): 777–785.
18. Balasubramaniam, P., and V. P. Ananthi. "Segmentation of nutrient deficiency in incomplete crop images using intuitionistic fuzzy C-means clustering algorithm." Nonlinear Dynamics 83.1-2 (2016): 849–866.
19. J. C. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algorithms", New York, Plenum, 1981.
20. Gomathi, M., and P. Thangaraj. "A parameter based modified fuzzy possibilistic c-means clustering algorithm for lung image segmentation." Global Journal of Computer Science and Technology (2010).
21. Saad, Mohamed Fadhel, and Adel M. Alimi. "Modified fuzzy possibilistic c-means." Proceedings of the international multi-conference of engineers and computer scientists. Vol. 1. 2009.

22. Ganesan, P., and V. Rajini. “A method to segment color images based on modified fuzzy-possibilistic-c-means clustering algorithm.” Recent Advances in Space Technology Services and Climate Change 2010 (RSTS & CC-2010). IEEE, 2010.
23. Praneesh, M., and R. Jaya Kumar. “Novel approach for color based comic image segmentation for extraction of text using modify fuzzy possibilistic c-means clustering algorithm.” Int J Comput Appl IPRC 1 (2012): 16–18.
24. S.C. Chen, D.Q. Zhang, Robust image segmentation using FCM with spatial constraints based on new kernel-induced distance measure. IEEE Trans. Syst. Man Cybern. 34, 1907–1916 (2004)
25. E.A. Zanaty, S. Aljahdali, N. Debnath, A kernelized fuzzy c-means algorithm for automatic magnetic resonance image segmentation. J. Comput. Methods Sci. Eng. 9, 123–136 (2009).
26. Kannan, S. R., et al. “Effective Kernel-Based Fuzzy Clustering Systems in Analyzing Cancer Database.” Data-Enabled Discovery and Applications 2.1 (2018): 5.
27. Kalam, Rehna, Ciza Thomas, and M. Abdul Rahiman. “GAUSSIAN KERNEL BASED FUZZY CMeans CLUSTERING ALGORITHM FOR IMAGE SEGMENTATION.” Comput. Sci. Inf. Technol (2016): 47–56.
28. Wang, Qiuping, et al. “Kernel-based fuzzy C-means clustering based on fruit fly optimization algorithm.” 2017 International Conference on Grey Systems and Intelligent Services (GSIS). IEEE, 2017.
29. Torshizi, Abolfazl Doostparast, and Mohammad Hossein Fazel Zarandi. “A new cluster validity measure based on general type-2 fuzzy sets: application in gene expression data clustering.” Knowledge-Based Systems 64 (2014): 81–93.
30. Begum, Shahin Ara, and O. Mema Devi. “A rough type-2 fuzzy clustering algorithm for MR image segmentation.” International Journal of Computer Applications 54.4 (2012).
31. Linda, Ondrej, and Milos Manic. “General type-2 fuzzy c-means algorithm for uncertain fuzzy clustering.” IEEE Transactions on Fuzzy Systems 20.5 (2012): 883–897.
32. Hwang, C., Rhee, F.C.H.: Uncertain Fuzzy Clustering: Interval Type-2 Fuzzy Approach to C-Means. IEEE Trans. On Fuzzy Systems 15(1), 107–120 (2007).
33. Dang, Trong Hop, Long Thanh Ngo, and Witold Pedrycz. “Interval Type-2 fuzzy C-Means approach to collaborative clustering.” 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, 2015.
34. Nguyen, Thanh, et al. “Medical data classification using interval type-2 fuzzy logic system and wavelets.” Applied Soft Computing 30 (2015): 812–822.
35. Y.Zheng,B.Jeon,D. Xu, et al., Image segmentation by generalized hierarchical fuzzy C-means algorithm, J. Intell. Fuzzy Syst.: Appl. Eng. Technol. 28(2) (2015)961–973.
36. Chen, Yunjie, et al. “An improved anisotropic hierarchical fuzzy c-means method based on multivariate student t-distribution for brain MRI segmentation.” Pattern Recognition 60 (2016): 778–792.
37. Bordogna, Gloria, and Gabriella Pasi. “Hierarchical-hyperspherical divisive fuzzy c-means (h2d-fcm) clustering for information retrieval.” 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. Vol. 1. IEEE, 2009.
38. Chen, Yunjie, et al. “Non-local-based spatially constrained hierarchical fuzzy C-means method for brain magnetic resonance imaging segmentation.” IET Image Processing 10.11 (2016): 865–876.
39. Davé, Rajesh N., and Sumit Sen. “Robust fuzzy clustering of relational data.” IEEE Transactions on Fuzzy Systems 10.6 (2002): 713–727.
40. Wan, Ling, et al. “A robust fuzzy c-means algorithm based on Bayesian nonlocal spatial information for SAR image segmentation.” IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 11.3 (2018): 896–906.
41. Cimino, Mario GCA, et al. “On the Noise Distance in Robust Fuzzy C-Means.” International Conference on Computational Intelligence. 2004.
42. Lei, Tao, et al. “Significantly fast and robust fuzzy c-means clustering algorithm based on morphological reconstruction and membership filtering.” IEEE Transactions on Fuzzy Systems 26.5 (2018): 3027–3041.

43. Pedrycz, Witold, and James Waletzky. “Fuzzy clustering with partial supervision.” IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 27.5 (1997): 787–795.
44. Lai, Daphne Teck Ching, and Jonathan M. Garibaldi. “A preliminary study on automatic breast cancer data classification using semi-supervised fuzzy c-means.” International Journal of Biomedical Engineering and Technology 13.4 (2013): 303–322.
45. Liliana, Dewi Yanti, M. Rahmat Widyanto, and T. Basaruddin. “Human emotion recognition based on active appearance model and semi-supervised fuzzy C-means.” 2016 international conference on advanced computer science and information systems (ICACSIS). IEEE, 2016.
46. Guorui, Feng, Zou Xinguo, and Wu Jian. “Intrusion detection based on the semi-supervised Fuzzy C-Means clustering algorithm.” 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet). IEEE, 2012.
47. Bouyer, Asgarali, Abdolreza Hatamlou, and Mohammad Masdari. “A new approach for decreasing energy in wireless sensor networks with hybrid LEACH protocol and fuzzy C-means algorithm.” IJCNDS 14.4 (2015): 400–412.
48. Li, Peng, et al. “A privacy-preserving high-order neuro-fuzzy c-means algorithm with cloud computing.” Neurocomputing 256 (2017): 82–89.
49. Pandeeswari, N., and Ganesh Kumar. “Anomaly detection system in cloud environment using fuzzy clustering-based ANN.” Mobile Networks and Applications 21.3 (2016): 494–505.
50. Lotfan, Saeed, et al. “Bearing fault detection using fuzzy C-means and hybrid C-means-subtractive algorithms.” 2015 IEEE international conference on fuzzy systems (FUZZ-IEEE). IEEE, 2015.
51. Zhao, Ruonan, Lize Gu, and Xiaoning Zhu. “Combining Fuzzy C-Means Clustering with Fuzzy Rough Feature Selection.” Applied Sciences 9.4 (2019): 679.
52. Karimi Ghaleh Jough, Fooad, and Serhan Şensoy. “Steel Moment-Resisting Frame Reliability via the Interval Analysis by FCM-PSO Approach considering Various Uncertainties.” Journal of Earthquake Engineering 24.1 (2020): 109–128.
53. Chu, Haibo, Jiahua Wei, and Wenyan Wu. “Streamflow prediction using LASSO-FCM-DBN approach based on hydro-meteorological condition classification.” Journal of Hydrology 580 (2020): 124253. (7)
54. Vandarkuzhali, T., and C. S. Ravichandran. “Detection of fovea region in retinal images using optimisation-based modified FCM and ARMD disease classification with SVM.” International Journal of Biomedical Engineering and Technology 32.1 (2020): 83–107. (8)
55. Majdi, Abbas, and Morteza Beiki. “Applying evolutionary optimization algorithms for improving fuzzy C-mean clustering performance to predict the deformation modulus of rock mass.” International Journal of Rock Mechanics and Mining Sciences 113 (2019): 172–182. (9)
56. Anter, Ahmed M., and Aboul Ella Hassenian. “CT liver tumor segmentation hybrid approach using neutrosophic sets, fast fuzzy c-means and adaptive watershed algorithm.” Artificial intelligence in medicine 97 (2019): 105–117. (10)
57. Hu, Yongtao, et al. “A new method of wind turbine bearing fault diagnosis based on multi-masking empirical mode decomposition and fuzzy c-means clustering.” Chinese Journal of Mechanical Engineering 32.1 (2019): 1–12. (11)
58. Kumar, Dhirendra, et al. “A modified intuitionistic fuzzy c-means clustering approach to segment human brain MRI image.” Multimedia Tools and Applications 78.10 (2019): 12663–12687.
59. Zhang, Minxia, et al. “A hybrid biogeography-based optimization and fuzzy C-means algorithm for image segmentation.” Soft computing 23.6 (2019): 2033–2046.
60. Zeraatpisheh, Mojtaba, et al. “Disaggregating and updating a legacy soil map using DSMART, fuzzy c-means and k-means clustering algorithms in Central Iran.” Geoderma 340 (2019): 249–258.
61. Azzaoui, H., I. Mansouri, and B. Elkihel. “Methylcyclohexane Continuous Distillation Column Fault Detection Using Stationary Wavelet Transform & Fuzzy C-means.” Materials Today: Proceedings 13 (2019): 597–606.

62. Wu, Tong, et al. “Modified fuzzy clustering with segregated cluster centroids.” Neurocomputing 361 (2019): 10–18.
63. Rezaee, Mustafa Jahangoshai, Mehrdad Jozmaleki, and Mahsa Valipour. “Integrating dynamic fuzzy C-means, data envelopment analysis and artificial neural network to online prediction performance of companies in stock exchange.” Physica A: Statistical Mechanics and its Applications 489 (2018): 78–93. (12)
64. Nayak, Janmenjoy, et al. “A hybrid elicit teaching learning based optimization with fuzzy c-means (ETLBO-FCM) algorithm for data clustering.” Ain Shams Engineering Journal 9.3 (2018): 379–393. (13)
65. Alsmirat, Mohammad A., et al. “Accelerating compute intensive medical imaging segmentation algorithms using hybrid CPU-GPU implementations.” Multimedia Tools and Applications 76.3 (2017): 3537–3555. (14)
66. Nayak, Janmenjoy, et al. “Hybrid chemical reaction based metaheuristic with fuzzy c-means algorithm for optimal cluster analysis.” Expert Systems with Applications 79 (2017): 282–295. (15)
67. Ouma, Yashon O., and M. Hahn. “Pothole detection on asphalt pavements from 2D-colour pothole images using fuzzy c-means clustering and morphological reconstruction.” Automation in Construction 83 (2017): 196–211. (16)
68. Fatehi, Moslem, and Hooshang H. Asadi. “Application of semi-supervised fuzzy c-means method in clustering multivariate geochemical data, a case study from the Dalli Cu-Au porphyry deposit in central Iran.” Ore Geology Reviews 81 (2017): 245–255.
69. Kumar, Ajit, Dharmender Kumar, and S. K. Jarial. “A hybrid clustering method based on improved artificial bee colony and fuzzy C-Means algorithm.” Int. J. Artif. Intell 15.2 (2017): 24–44.
70. Rubio, Elid, et al. “An extension of the fuzzy possibilistic clustering algorithm using type-2 fuzzy logic techniques.” Advances in Fuzzy Systems 2017 (2017).
71. Kisi, Ozgur, and Mohammad Zounemat-Kermani. “Suspended sediment modeling using neuro-fuzzy embedded fuzzy c-means clustering technique.” Water Resources Management 30.11 (2016): 3979–3994. (17)
72. Bai, Chunguang, Dileep Dhavale, and Joseph Sarkis. “Complex investment decisions using rough set and fuzzy c-means: An example of investment in green supply chains.” European journal of operational research 248.2 (2016): 507–521. (18)
73. Chattaraj, Arnab, and Arpita Das. “Mammographie image segmentation using kernel based FCM clustering approach.” 2016 International Conference on Computer, Electrical & Communication Engineering (ICCECE). IEEE, 2016.
74. Ding, Yi, and Xian Fu. “Kernel-based fuzzy c-means clustering algorithm based on genetic algorithm.” Neurocomputing 188 (2016): 233–238.
75. Fathabadi, Hassan. “Power distribution network reconfiguration for power loss minimization using novel dynamic fuzzy c-means (dFCM) clustering based ANN approach.” International Journal of Electrical Power & Energy Systems 78 (2016): 96–107.
76. Ahmmed, Rasel, and Md Foisal Hossain. “Tumor detection in brain MRI image using template based K-means and Fuzzy C-means clustering algorithm.” 2016 International Conference on Computer Communication and Informatics (ICCCI). IEEE, 2016.
77. Keskin, Gülşen Aydın. “Using integrated fuzzy DEMATEL and fuzzy C: means algorithm for supplier evaluation and selection.” International Journal of Production Research 53.12 (2015): 3586–3602. (19)
78. Abdel-Maksoud, Eman, Mohammed Elmogy, and Rashid Al-Awadi. “Brain tumor segmentation based on a hybrid clustering technique.” Egyptian Informatics Journal 16.1 (2015): 71–81. (20)
79. Mai, Sinh Dinh, and Long Thanh Ngo. “Interval type-2 Fuzzy C-means clustering with spatial information for land-cover classification.” Asian Conference on Intelligent Information and Database Systems. Springer, Cham, 2015.

80. Tang, Jinjun, et al. “A hybrid approach to integrate fuzzy C-means based imputation method with genetic algorithm for missing traffic volume data estimation.” Transportation Research Part C: Emerging Technologies 51 (2015): 29–40.
81. Gupta, Deep, R. S. Anand, and Barjeev Tyagi. “A hybrid segmentation method based on Gaussian kernel fuzzy clustering and region based active contour model for ultrasound medical images.” Biomedical Signal Processing and Control 16 (2015): 98–112.
82. Singh, Amritpal. “Detection of brain tumor in MRI images, using combination of fuzzy c-means and SVM.” 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN). IEEE, 2015.
83. Izakian, Hesam, Witold Pedrycz, and Iqbal Jamal. “Fuzzy clustering of time series data using dynamic time warping distance.” Engineering Applications of Artificial Intelligence 39 (2015): 235–244.
84. Silva Filho, Telmo M., et al. “Hybrid methods for fuzzy clustering based on fuzzy c-means and improved particle swarm optimization.” Expert Systems with Applications 42.17–18 (2015): 6315–6328.
85. Mekhmoukh, Abdenour, and Karim Mokrani. “Improved Fuzzy C-Means based Particle Swarm Optimization (PSO) initialization and outlier rejection with level set methods for MR brain image segmentation.” Computer methods and programs in biomedicine 122.2 (2015): 266–281.
86. Gupta, Anshul, Hirdesh Shivhare, and Shalki Sharma. “Recommender system using fuzzy c-means clustering and genetic algorithm based weighted similarity measure.” 2015 International Conference on Computer, Communication and Control (IC4). IEEE, 2015.
87. Stetco, Adrian, Xiao-Jun Zeng, and John Keane. “Fuzzy C-means++: Fuzzy C-means with effective seeding initialization.” Expert Systems with Applications 42.21 (2015): 7541–7548.
88. Son, Le Hoang. “DPFCM.” Expert Systems with Applications: An International Journal 42.1 (2015): 51–66.
89. Manimala, K., Indra Getzy David, and K. Selvi. “A novel data selection technique using fuzzy C-means clustering to enhance SVM-based power quality classification.” Soft Computing 19.11 (2015): 3123–3144.
90. Zhang, Liyong, et al. “Fuzzy c-means clustering of incomplete data based on probabilistic information granules of missing values.” Knowledge-Based Systems 99 (2016): 51–70.
91. Hassanien, Aboul Ella, Eid Emary, and Hossam M. Zawbaa. “Retinal blood vessel localization approach based on bee colony swarm optimization, fuzzy c-means and pattern search.” Journal of Visual Communication and Image Representation 31 (2015): 186–196.
92. Mansouri, Iman, et al. “Evaluation of peak and residual conditions of actively confined concrete using neuro-fuzzy and neural computing techniques.” Neural Computing and Applications 29.3 (2018): 873–888.
93. Mahela, Om Prakash, and Abdul Gafoor Shaik. “Recognition of power quality disturbances using S-transform based ruled decision tree and fuzzy C-means clustering classifiers.” Applied Soft Computing 59 (2017): 243–257.
94. Rajaby, Elias, Seyed Mohammad Ahadi, and Hassan Aghaeinia. “Robust color image segmentation using fuzzy c-means with weighted hue and intensity.” Digital Signal Processing 51 (2016): 170–183.
95. John, Vijay, et al. “Pedestrian detection in thermal images using adaptive fuzzy C-means clustering and convolutional neural networks.” 2015 14th IAPR International Conference on Machine Vision Applications (MVA). IEEE, 2015.
96. Agarwal, Pankhuri, et al. “A combination of bias-field corrected fuzzy c-means and level set approach for brain MRI image segmentation.” 2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI). IEEE, 2015.
97. Verma, Hanuman, R. K. Agrawal, and Aditi Sharan. “An improved intuitionistic fuzzy c-means clustering algorithm incorporating local information for brain image segmentation.” Applied Soft Computing 46 (2016): 543–557.

98. Hien, Nguyen Mong, Nguyen Thanh Binh, and Ngo Quoc Viet. "Edge detection based on Fuzzy C Means in medical image processing system." 2017 International Conference on System Science and Engineering (ICSSE). IEEE, 2017.
99. Syed, Zaheeruddin, and Dr K. Suganthi. "A Combined Approach of Retinex & Spatial Kernel Fuzzy C-Means Clustering for Detection of Oil Spills in Satellite Imagery." Available at SSRN 3358049 (2019).
100. Hu, Guang, and Zhenbin Du. "Adaptive kernel-based fuzzy c-means clustering with spatial constraints for image segmentation." International Journal of Pattern Recognition and Artificial Intelligence 33.01 (2019): 1954003.
101. Bilenia, Aniket, et al. "Brain tumor segmentation with skull stripping and modified fuzzy C-means." Information and Communication Technology for Intelligent Systems. Springer, Singapore, 2019. 229–237
102. Nida, Nudrat, et al. "Melanoma lesion detection and segmentation using deep region based convolutional neural network and fuzzy C-means clustering." International journal of medical informatics 124 (2019): 37–48.
103. Alsmadi, Mutasem K. "A hybrid Fuzzy C-Means and Neutrosophic for jaw lesions segmentation." Ain Shams Engineering Journal 9.4 (2018): 697–706.
104. Noh, Yohwan, et al. "Automatic crack detection on concrete images using segmentation via fuzzy C-means clustering." 2017 International Conference on Applied System Innovation (ICASI). IEEE, 2017.
105. Shedthi, B. Shabari, Surendra Shetty, and M. Siddappa. "Implementation and comparison of K-means and fuzzy C-means algorithms for agricultural data." 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT). IEEE, 2017.
106. Shang, Ronghua, et al. "A spatial fuzzy clustering algorithm with kernel metric based on immune clone for SAR image segmentation." IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 9.4 (2016): 1640–1652.
107. Marrone, Stefano, et al. "Breast segmentation using Fuzzy C-Means and anatomical priors in DCE-MRI." 2016 23rd International Conference on Pattern Recognition (ICPR). IEEE, 2016.
108. Feng, Chaolu, Dazhe Zhao, and Min Huang. "Image segmentation using CUDA accelerated non-local means denoising and bias correction embedded fuzzy c-means (BCEFCM)." Signal Processing 122 (2016): 164–189.
109. Adhikari, Sudip Kumar, et al. "Conditional spatial fuzzy C-means clustering algorithm for segmentation of MRI images." Applied soft computing 34 (2015): 758–769.
110. Li, Yaoguo, and Jiajia Sun. "3D magnetization inversion using fuzzy c-means clustering with application to geology differentiation." Geophysics 81.5 (2016): J61-J78.
111. Demircan, Semiye, and Humar Kahramanli. "Application of fuzzy C-means clustering algorithm to spectral features for emotion classification from speech." Neural Computing and Applications 29.8 (2018): 59–66.
112. Phu, Vo Ngoc, et al. "Fuzzy C-means for English sentiment classification in a distributed system." Applied Intelligence 46.3 (2017): 717–738.
113. Yang, Miin-Shen, and Yessica Nataliani. "Robust-learning fuzzy c-means clustering algorithm with unknown number of clusters." Pattern Recognition 71 (2017): 45–59.
114. Jiekang, Wu, et al. "Risk early warning method for distribution system with sources-networks-loads-vehicles based on fuzzy C-mean clustering." Electric Power Systems Research 180 (2020): 106059.
115. Ranjbarzadeh, Ramin, and Soroush Baseri Saadi. "Automated liver and tumor segmentation based on concave and convex points using fuzzy c-means and mean shift clustering." Measurement 150 (2020): 107086.
116. Silva, Gustavo RL, et al. "A fuzzy data reduction cluster method based on boundary information for large datasets." Neural Computing and Applications (2019): 1–10.
117. Zeng, Shan, et al. "A study on multi-kernel intuitionistic fuzzy C-means clustering with multiple attributes." Neurocomputing 335 (2019): 59–71.
118. Palani, D., and K. Venkatalakshmi. "An IoT based predictive modelling for predicting lung cancer using fuzzy cluster based segmentation and classification." Journal of medical systems 43.2 (2019): 21.

119. Sharma, Ram Prakash, and Somnath Dey. “Two-stage quality adaptive fingerprint image enhancement using Fuzzy C-means clustering based fingerprint quality analysis.” Image and Vision Computing 83 (2019): 1–16.
120. Zhao, Qiang, et al. “A new PV array fault diagnosis method using fuzzy C-mean clustering and fuzzy membership algorithm.” Energies 11.1 (2018): 238.
121. Khan, Muhammad Jaleed, et al. “Automated forgery detection in multispectral document images using fuzzy clustering.” 2018 13th IAPR International Workshop on Document Analysis Systems (DAS). IEEE, 2018.
122. Haldar, Nur Al Hasan, et al. “Arrhythmia classification using Mahalanobis distance based improved Fuzzy C-Means clustering for mobile health monitoring systems.” Neurocomputing 220 (2017): 221–235.
123. Benmouiza, Khalil, Mohammed Tadj, and Ali Cheknane. “Classification of hourly solar radiation using fuzzy c-means algorithm for optimal stand-alone PV system sizing.” International journal of electrical power & energy systems 82 (2016): 233–241.
124. Lin, Phen-Lan, Po-Whei Huang, and Chun-Hung Kuo. “A noise-and size-insensitive integrity-based fuzzy c-means algorithm for image segmentation.” 2015 IEEE International Conference on Automation Science and Engineering (CASE). IEEE, 2015.
125. Yang, Miin-Shen, and Yi-Cheng Tian. “Bias-correction fuzzy clustering algorithms.” Information Sciences 309 (2015): 138–162.
126. Rai, Khushnandan, Varun Bajaj, and Anil Kumar. “Hilbert-Huang transform based classification of sleep and wake EEG signals using fuzzy c-means algorithm.” 2015 International Conference on Communications and Signal Processing (ICCSP). IEEE, 2015.
127. Kisi, Ozgur. “Streamflow forecasting and estimation using least square support vector regression and adaptive neuro-fuzzy embedded fuzzy c-means clustering.” Water Resources Management 29.14 (2015): 5109–5127.
128. Jagannath, M., and K. Adalarasu. “Diagnosis of diabetic retinopathy from fundus image using fuzzy c-means clustering algorithm.” IIOAB Journal. ISSN (2015): 0976–3104.
129. Khan, Javed, et al. “Segmentation of acne lesion using fuzzy C-means technique with intelligent selection of the desired cluster.” 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2015.
130. Rustam, Zuherman, and Durrabida Zahras. “Comparison between support vector machine and fuzzy c-means as classifier for intrusion detection system.” Journal of Physics: Conference Series. Vol. 1028. No. 1. IOP Publishing, 2018.
131. Zhang, Zhongxing, and Baoping Gu. “Intrusion detection network based on fuzzy c-means and particle swarm optimization.” Proceedings of the 6th International Asia Conference on Industrial Engineering and Management Innovation. Atlantis Press, Paris, 2016.
132. Rustam, Z., and J. Maharani. “Intrusion detection system model using Fuzzy Kernel C-Means and Laplacian Score feature selection.” Journal of Physics: Conference Series. Vol. 1442. No. 1. IOP Publishing, 2020.
133. Zhang, Zhiyou, and Peisheng Pan. “A Hybrid Intrusion Detection Method Based on Improved Fuzzy C-Means and Support Vector Machine.” 2019 International Conference on Communications, Information System and Computer Engineering (CISCE). IEEE, 2019.
134. Duan, Liyu, and Youan Xiao. “An Intrusion Detection Model Based on Fuzzy C-means Algorithm.” 2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC). IEEE, 2018.
135. Kumar, Kaushal. “An efficient network intrusion detection system based on fuzzy C-means and support vector machine.” 2016 International Conference on Computer, Electrical & Communication Engineering (ICCECE). IEEE, 2016.
136. Rustam, Zuherman, and Aini Suri Talita. “Fuzzy kernel c-means algorithm for intrusion detection systems.” Journal of Theoretical and Applied Information Technology 81.1 (2015): 161.
137. Sampat, Richa, and Shilpa Sonawani. “Network intrusion detection using dynamic fuzzy c means clustering.” Network 2.4 (2015)

Clustering in Streams

Charu C. Aggarwal

1 Introduction

Streaming data has become particularly popular in the big data age; it is generated by a wide variety of real-world applications [1], such as using the credit card or the phone. The amount of data is so large that it is sometimes hard to retain all of it within the required storage constraints. In other cases, when sufficient storage is available, it is hard to utilize this large volume of data for mining applications. Clustering is a natural approach to summarize the data in a way so that it can be leveraged in an efficient way in various data-centric applications. The basic principle is that one does not need the raw data for most applications. In most cases, a mathematical model is constructed that summarizes the data stream, and this model can be used in many applications. Clustering is one of the many models that is used in the streaming setting.

The problem of clustering has been widely studied in the data mining and machine learning communities for more than six decades. Classical methods include the use of the k-means algorithm, k-medians algorithm, density-based methods, probabilistic clustering methods, and spectral methods. A detailed discussion of different kinds of clustering methods may be found in [44, 46]. However, many of the classical methods do not work very well in the streaming setting. The specific reasons for the additional challenges associated with data streams are as follows:

- Data streams have massive volume that continuously arrives over long periods in term. This precludes the use of conventional methods based on storing the data for later use. This means that the data needs to be processed in a *single*

C. C. Aggarwal (✉)
IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
e-mail: charu@us.ibm.com

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_13

pass, in which all the summary information required for the clustering process needs to be stored and maintained.
- Data stream continuously evolve over time, and this phenomenon is referred to as *concept drift* [13]. Therefore, the clustering models need to be sensitive to this type of concept drift.
- Different domains of data may pose different challenges to data stream clustering. For example, in a *massive domain* of discrete attributes, it may not be possible to store summary representations of the clusters effectively without increasing the computational complexity of the problem significantly. Therefore, space-efficient methods need to be designed for massive-domain clustering of data streams.
- In some specialized domains, such as graphs, it is hard to create summarized representations of the clusters. This process requires specialized randomized techniques such as sketches for summarization.

While scalability is critical for stream processing, it is the evolving nature of the data that causes greater challenges. This type of evolving nature is also referred to as *temporal locality*. Scalability issues have been studied in the context of very large data sets [32, 38, 53, 65], and many of these ideas have been generalized to data streams. However, the stream setting is more challenging because large data sets do allow multiple passes (within reason). Furthermore, one does not have to worry about the evolving nature of the data when the data set is large.

This chapter is organized as follows. The next section discusses representative-based methods. Density-based methods are discussed in Sect. 3. Section 4 discusses probabilistic algorithms for clustering data streams. High-dimensional streaming algorithms are introduced in Sect. 5. Discrete and categorical stream clustering is discussed in Sect. 6. As discussed in Sect. 7, these methods can also be generalized to text streams. Challenging settings like graphs and distributed scenarios are discussed in Sect. 8. A summary is given in Sect. 9.

2 Representative-Based Methods

In representative-based methods, a set of cluster representatives is designated up front, and these are then utilized in order to build clusters around these representative points. Examples of such methods include the k-means and k-medians-based methods. The cluster membership of the points is defined by assigning them to their closest representative. An algorithm like k-means requires multiple passes over the data. This is not a feasible approach in the streaming setting, where using a single pass is a *hard* requirement. Therefore, further changes are needed to such methods. This section will discuss these methods. Another common challenge in representative-based algorithms is that the number of clusters may evolve over time. Some algorithms like [37] keep the number of clusters fixed, whereas others like [14, 28] allow the number of clusters to evolve over time.

2.1 The STREAM Algorithm

The *STREAM* framework [37, 55] breaks the stream into *chunks* of smaller size, which are denoted by $\mathcal{D}_1 \ldots \mathcal{D}_r \ldots$. Each chunk is constrained to contain at most m data points based on a predefined memory budget, and an arbitrary algorithm can be used for processing a chunk. The methods in [37, 55] use k-medians style algorithms on the individual chunks.

In this approach, a set $\mathcal{S}$ of k representatives is selected from each chunk $\mathcal{D}_i$, so that each point in $\mathcal{D}_i$ is assigned to its closest representatives. The goal is to pick the representatives in such a way, so as to minimize the distance of the assigned data points from these representatives. For a set of m data points $\overline{X_1} \ldots \overline{X_m}$ in $\mathcal{S}$, a set of k representatives $\mathcal{Y} = \overline{Y_1} \ldots \overline{Y_k}$ is selected. Each point is assigned to its closest representative. After the first chunk has been processed, we now have a set of k representatives, which are stored away. The number of points assigned to each representative is stored as a "weight" for that representative. Such representatives are considered *level-1* representatives. The next chunk is independently processed in order to find its k optimal median representatives. Thus, at the end of processing the second chunk, we will have $2 \cdot k$ such representatives. Thus, the memory requirement for storing the representatives also increases with time, and after processing r chunks, we will have a total of $r \cdot k$ representatives. When the number of representatives exceeds m, a second level of clustering is applied to this set of $r \cdot k$ points, except that the stored weights on the representatives are also used in the clustering process. The resulting representatives are stored as *level-2* representatives. In general, when the number of representatives of level-p reaches m, they are converted to k level-$(p + 1)$ representatives. Thus, the process will result in increasing the number of representatives of all levels, though the number of representatives in higher levels will increase exponentially slower than those in the lower levels. At the end of processing the entire data stream (or when a specific need for the clustering result arises), all remaining representatives of different levels are clustered together in one final application of the same clustering subroutine.

Clearly, the choice of the particular algorithm which is used for the clustering subroutine will have an effect on the quality of the solution. It has been shown in [55] that the k-medians approach works well because the quality of the solution obeys a similar approximation factor.

Lemma 1 *Let the subroutine used for k-medians clustering in the* STREAM *algorithm have an approximation factor of c. Then, the* STREAM *algorithm will have an approximation factor of no worse than $5 \cdot c$.*

The STREAM algorithm is not very sensitive to data stream evolution. In such cases, the *CluStream* algorithm is able to provide significantly better insights at different levels of temporal granularity.

2.2 Micro-clustering: The Big Picture

The micro-clustering framework is designed to be able to simultaneously maintain the results of clusters over multiple time horizons. In many cases, an analyst may wish to determine the clusters at a previous moment in time and compare them to the current clusters. Therefore, a natural design to stream clustering would be separate out the process into an online micro-clustering component and an offline macro-clustering component. The online micro-clustering component requires a very efficient process for storage of appropriate summary statistics in a fast data stream. The offline component uses these summary statistics for user-centric exploration.

We start by introducing the notations and definitions. The data stream consists of a set of multidimensional records $\overline{X}_1 \ldots \overline{X}_k \ldots$ arriving at time stamps $T_1 \ldots T_k \ldots$. Each $\overline{X_i}$ is a multidimensional record containing d dimensions which are denoted by $\overline{X_i} = (x_i^1 \ldots x_i^d)$. The micro-clustering framework is designed to capture summary information about the data stream in order to facilitate clustering and analysis over different time horizons. This summary information is defined by the following structures:

- **Micro-clusters:** These structures contain statistical information about the data locality in terms of micro-clusters. These micro-clusters are defined as temporal extensions of the *cluster feature vector* [65]. The additivity property of the micro-clusters makes them a natural choice for the data stream problem.
- **Pyramidal Time Frame:** The micro-clusters are stored at snapshots in time which follow a pyramidal pattern. This pattern provides an effective trade-off between the storage requirements and the ability to recall summary statistics from different time horizons.

We start by providing a definition of micro-clusters.

2.2.1 Defining Micro-clusters

The concept of micro-clustering is similar to that in the BIRCH algorithm [65], except that it contains some additional information to handle the temporal aspects of the data:

Definition 1 A micro-cluster for a set of d-dimensional points $X_{i_1} \ldots X_{i_n}$ with time stamps $T_{i_1} \ldots T_{i_n}$ is the $(2 \cdot d + 3)$ tuple $(\overline{CF2^x}, \overline{CF1^x}, CF2^t, CF1^t, n)$, wherein $\overline{CF2^x}$ and $\overline{CF1^x}$ each correspond to a vector of d entries. The different components of the micro-cluster are as follows:

- For each dimension, the sum of the squares of the data values is maintained in $\overline{CF2^x}$. Thus, $\overline{CF2^x}$ contains d values. The pth entry of $\overline{CF2^x}$ is equal to $\sum_{j=1}^{n} (x_{i_j}^p)^2$.
- For each dimension, the sum of the data values is maintained in $\overline{CF1^x}$. Thus, $\overline{CF1^x}$ contains d values. The pth entry of $\overline{CF1^x}$ is equal to $\sum_{j=1}^{n} x_{i_j}^p$.

- The sum of the squares of the time stamps $T_{i_1} \ldots T_{i_n}$ is maintained in $CF2^t$.
- The sum of the time stamps $T_{i_1} \ldots T_{i_n}$ is maintained in $CF1^t$.
- The number of data points is maintained in n.

The data stream clustering algorithm proposed in [14] can generate approximate clusters in any user-specified length of history from the current instant. This is achieved by storing the micro-clusters at particular moments in the stream which are referred to as *snapshots*. At the same time, the current snapshot of micro-clusters is always maintained by the algorithm. Consider, for example, the case when the current clock time is t_c and the user wishes to find clusters in the stream based on a history of length h. Then, the macro-clustering algorithm will use some of the additive properties of the micro-clusters stored at snapshots t_c and $(t_c - h)$ in order to find the higher level clusters in a history or *time horizon* of length h. Of course, since it is not possible to store the snapshots at each and every moment in time, it is important to choose particular instants of time at which it is possible to store the state of the micro-clusters so that clusters in any user-specified time horizon $(t_c - h, t_c)$ can be approximated. This was achieved in [14] with the use of the concept of a pyramidal time frame.

2.2.2 Pyramidal Time Frame

The pyramidal time frame arranges the snapshots of the stream based on their level of recency. Snapshots are classified into different *orders* which can vary from 1 to $\log(T)$, where T is the clock time elapsed since the beginning of the stream. The order of a particular class of snapshots defines the level of granularity in time at which the snapshots are maintained. The snapshots of different order are maintained as follows:

- Snapshots of the ith order occur at time intervals of α^i, where α is an integer and $\alpha \geq 1$. Specifically, each snapshot of the ith order is taken at a moment in time when the clock value is exactly divisible by α^i.
- At any given moment in time, only the last $\alpha^l + 1$ snapshots of order i are stored.

The above definition allows for considerable redundancy in storage of snapshots. For example, the clock time of 8 is divisible by 2^0, 2^1, 2^2, and 2^3 (where $\alpha = 2$). Therefore, the state of the micro-clusters at a clock time of 8 simultaneously corresponds to order 0, order 1, order 2, and order 3 snapshots. From an implementation point of view, a snapshot needs to be maintained only once. The following observations are true:

- For a data stream, the maximum order of any snapshot stored at T time units since the beginning of the stream mining process is $\log_\alpha(T)$.
- For a data stream, the maximum number of snapshots maintained at T time units since the beginning of the stream mining process is $(\alpha^l + 1) \cdot \log_\alpha(T)$.
- For any user-specified time window of h, at least one stored snapshot can be found within $(1 + 1/\alpha^{l-1})$ units of the current time.

The second of the two results has been formally proven in [14]:

Lemma 2 *Let h be a user-specified time horizon, t_c be the current time, and t_s be the time of the last stored snapshot of any order just before the time $t_c - h$. Then, $t_c - t_s \leq (1 + 1/\alpha^{l-1}) \cdot h$.*

For larger values of l, the time horizon can be approximated as closely as desired. For example, by choosing $l = 10$, it is possible to approximate any time horizon within 0.2%, while a total of only $(2^{10} + 1) \cdot \log_2(100 * 365 * 24 * 60 * 60) \approx 32343$ snapshots are required for 100 years. Since historical snapshots can be stored on disk and only the current snapshot needs to be maintained in main memory, this requirement is quite feasible from a practical point of view. It is also possible to specify the pyramidal time window in accordance with user preferences corresponding to particular moments in time such as beginning of calendar years, months, and days.

2.2.3 How Micro-clusters Are Created in Real Time

The goal of the online micro-clustering phase is to maintain statistics at a sufficiently high level of (temporal and spatial) granularity so that it can be effectively used by the offline components such as horizon-specific macro-clustering as well as evolution analysis. The algorithm works in an iterative fashion, by always maintaining a current set of micro-clusters. It is assumed that a total of q micro-clusters are stored at any moment by the algorithm. We will denote these micro-clusters by $\mathcal{M}_1 \ldots \mathcal{M}_q$. Associated with each micro-cluster i, we create a unique id whenever it is first created. If two micro-clusters are merged (as will become evident from the details of our maintenance algorithm), a *list* of ids is created in order to identify the constituent micro-clusters. The value of q is determined by the amount of main memory available in order to store the micro-clusters. Therefore, typical values of q are significantly larger than the natural number of clusters in the data but are also significantly smaller than the number of data points arriving in a long period of time for a massive data stream. These micro-clusters represent the current snapshot of clusters which change over the course of the stream as new points arrive. Their status is stored away on disk whenever the clock time is divisible by α^i for any integer i. At the same time, any micro-clusters of order r which were stored at a time in the past more remote than α^{l+r} units are deleted by the algorithm.

Whenever a new data point $\overline{X_{i_k}}$ arrives, the micro-clusters are updated in order to reflect the changes. Each data point either needs to be absorbed by a micro-cluster or needs to be put in a cluster of its own. The first preference is to absorb the data point into a currently existing micro-cluster. The distance of each data point to the micro-cluster centroids $\mathcal{M}_1 \ldots \mathcal{M}_q$ is determined. The distance value of the data point $\overline{X_{i_k}}$ to the centroid of the micro-cluster $\mathcal{M}_j$ is denoted by $dist(\mathcal{M}_j, \overline{X_{i_k}})$. Since the centroid of the micro-cluster is available in the cluster feature vector, this value can be computed relatively easily. This distance is used to compute the distance of the cluster $\mathcal{M}_p$ to the data point $\overline{X_{i_k}}$. In many cases, the point $\overline{X_{i_k}}$ does not naturally

belong to the cluster $\mathcal{M}_p$. In such cases, either the point is treated as an outlier, or a new cluster is created containing the data point. In some cases, related clusters may need to be merged. More details on how such points are identified and handled are provided in [14].

While the above process of updating is executed at the arrival of each data point, an additional process is executed at each clock time which is divisible by α^i for any integer i. At each such time, the current set of micro-clusters are stored on disk, together with their id list, and indexed by their time of storage. The least recent snapshot of order i is deleted, if $\alpha^l + 1$ snapshots of such order had already been stored on disk, and if the clock time for this snapshot is not divisible by α^{i+1}. In the latter case, the snapshot continues to be a viable snapshot of order $(i + 1)$. These micro-clusters can then be used to form higher level clusters or an evolution analysis of the data stream.

It should be pointed out that the micro-clustering model can be used in conjunction with fast indexing structures in order to allow *anytime* stream mining. This is particularly important in the context of data streams, since the stream speed is not known on an a priori basis. A particular approach along this direction is the *ClusTree* method [48], which allows the adaptation of the granularity of the cluster model to the stream speed. The broader principle is that it is possible to follow the anytime paradigm to spend as much (or as little) time as dynamically available to digest new events.

While the use of snapshots is a natural way for examining the evolving stream at a variety of different granularities, other methods are possible for capturing the evolution of the data stream by incorporating decay into the micro-clusters [15] or by using sliding windows in conjunction with an exponential histogram of the temporal cluster feature vector [68]. These methods are generally preferable if the level of evolution in the data stream is known in advance. These different methods have differing trade-offs between memory requirements and flexibility, but their goals are similar in terms of capturing the evolution of the data stream.

3 Density-Based Stream Clustering

Density-based methods [18, 31] construct a density profile of the data for clustering purposes. Typically, kernel density estimation methods [60] are used in order to construct a smooth density profile of the underlying data. Subsequently, the data is separated out into density-connected regions. These density-connected regions may be of different shapes and sizes. One of the advantages of density-based algorithms is that an implicit shape is not assumed for the clusters. For example, when Euclidean distance functions are used, it is always assumed that the clusters have spherical shapes. Similarly, the Manhattan metric assumes that the clusters are of a diamond shape. In density-based clustering, connected regions of high density may often have arbitrary shapes. Another aspect of density-based clustering is that it does not pre-decide the number of clusters. Rather, a threshold on the density is

used in order to determine the connected regions. Of course, this changes the nature of the parameter which needs to be presented to the algorithm (density threshold instead of the number of clusters), but it does not necessarily make the approach parameter-free.

The main challenge in the streaming scenario is to construct density-based algorithms, which can be efficiently executed in a single pass of the data, since the process of density estimation may be computationally intensive. There are two broad classes of techniques in this respect:

- The first class of techniques extends the micro-clustering technique to this case, by relaxing the constraint on the number of micro-clusters and imposing a constraint on the radius and "weight" of each micro-cluster. The dense regions are generated by connecting together the dense micro-clusters which satisfy a condition on connectivity similar to that in [31].
- The second class of techniques divides the data space into grids and then determines the dense grids. The dense regions in the data are reconstructed by piecing together the connected dense grids.

We will discuss both these classes of techniques in this section.

3.1 DenStream: Density-Based Micro-clustering

The *DenStream* algorithm [24] approach combines micro-clustering with a density estimation process for effective clustering. The first step is to define a *core object*, which is defined as an object, in the ϵ-neighborhood of which the weight of the data points is at least μ. A *density area* is defined as the union of the ϵ-neighborhoods of the core objects.

In the context of streaming data, it is difficult to determine these dense regions naturally. Therefore, dense regions of smaller granularity are defined in the form of core micro-clusters. A core micro-cluster is defined as a set of data points with weight at least μ and for which the radius of the micro-cluster about its center is less than ϵ. We note that the weight of a data point is based on a decay weighted function of the time that it last arrived. Therefore, if δt be the time since a data point arrived, its weight is given by

$$f(\delta t) = 2^{-\delta t} \tag{1}$$

Since the radius of the micro-cluster is constrained to be less than ϵ, it implies that the number of micro-clusters is much larger than the number of natural clusters in the data for small values of ϵ. At the same time, the number of core micro-clusters is much smaller than the number of points in the data stream since each cluster contains a weight of at least μ. We note that the key difference from the standard micro-clustering definition is that the number of micro-clusters is not constrained, though the radius of each micro-cluster is constrained. Thus, this

approach approaches a different parameter set to the underlying application. The core micro-cluster is also referred to as a c-micro-cluster.

One immediate observation is that when a micro-cluster is first formed by such an algorithm, it is unlikely to contain the requisite weight of data points required to be defined as a micro-cluster. Therefore, the concepts of *potential core micro-cluster* and *outlier micro-cluster* are defined in [24]. In the former case, the micro-cluster contains a weight of at least $\beta \cdot \mu$ (for some $\beta \in (0, 1)$), and in the latter case, it contains a weight less than $\beta \cdot \mu$. Furthermore, since the weights of points decay over time, a cluster may also change from being a p-micro-cluster to an o-micro-cluster, if sufficient data points are not added to it in order to compensate for the decay. Thus, during its lifecycle, a micro-cluster may move from being an outlier micro-cluster to being a potential core micro-cluster and finally to the stage of being a core micro-cluster. These two kinds of micro-clusters are referred to as p-micro-clusters and o-micro-clusters, respectively.

When a new data point arrives, the following can be the possibilities in terms of what is done with it:

- The first step is to try to insert it into a p-micro-cluster, as long as it is possible to do so, without violating the radius constraint.
- If the first step is not possible, the next step is to try to insert it into an o-micro-cluster, as long as this can be done without violating the radius constraint.
- If the first and second steps are not possible, then a new o-micro-cluster is created containing this data point.

One challenge of this approach is that the number of o-micro-clusters will increase with time, as new o-micro-clusters are created, and some of the p-micro-clusters decay back to o-micro-clusters. Therefore, from time to time, we purge some of the o-micro-clusters, which will low potential for becoming p-micro-clusters. The larger the time δt that has elapsed since the creation of an o-micro-cluster, the greater is the weight expected to be. Therefore, every T_p time periods, we prune all those micro-clusters, whose weight is less than the threshold $\psi(\delta t)$, where

$$\psi(\delta t) = \frac{2^{-\lambda \cdot (\delta t + T_p)} - 1}{2^{-\lambda \cdot (T_p)} - 1} \tag{2}$$

This process continues in order to maintain the micro-clusters dynamically. We note that the individual micro-clusters can be reconstructed into density-connected regions in order to create the final set of clusters of arbitrary shape. The overall approach for creating the clusters of arbitrary shape is discussed in detail in [24].

3.2 Grid-Based Streaming Algorithms

Grid-based methods are a class of density-based streaming algorithms, in which a grid structure is used in order to quantify the density at each point in the data. The

core idea is that the data is discretized into ranges along each dimension, and this also results in dividing the data into cells along different dimensions. The number of points in each cell defines the density of that cell. Then, the dense cells in the data can be aggregated in order to determine the dense regions for the clustering.

3.2.1 D-Stream Algorithm

A method called D-Stream for real-time density-based clustering of streaming data was proposed in [25]. The algorithm has many similarities with [31] in terms of trying to determine fine-grained regions of high density. The main difference at the conceptual level is that this is done with the use of grids rather than micro-clusters. As in the previous case, a decay function $f(\delta t(\overline{X}))$ is used to denote the weight of a point $\overline{X}$ since the time of its arrival $\delta t(\overline{X}, t_c)$ units ago from the current time t_c:

$$f(\delta t(\overline{X}, t_c)) = \mu^{-\delta t(\overline{X}, t_c)} \tag{3}$$

Here, we assume that $\mu > 1$, which is slightly different from the notations in [25]. We note that this decay function is identical to that proposed in [15, 31], by defining the relationship with respect to the parameter λ and assuming that $\mu = 2^{\lambda}$. Under the assumption of [25] that exactly one record arrives at each time stamp, it can be shown that the sum of the weights of all data points is no larger than $\mu/(\mu - 1)$.

The grids are defined by discretizing these ranges along each dimension. For the ith dimension, the discretization is performed into p_i different ranges along the ith dimension. This discretization naturally defines a total of $\eta = \prod_i p_i$ d-dimensional cells. For each cell S, its weight $W(S, t_c)$ is defined as the current time t_c as follows:

$$W(S, t_c) = \sum_{\overline{X} \in S} f(\delta t(\overline{X}), t_c) \tag{4}$$

We note that the grid is essentially analogous to the radius-constrained micro-cluster defined in the *DenStream* algorithm. Thus, as in the case of *DenStream*, the density of a grid is constantly changing over time. However, it is never necessary to update any decay-based statistics either in grids or in micro-clusters at each time instant [15, 25]. This is because all grids decay at the same proportional rate, and the update can be lazily performed only when the density value in the grid is updated with the addition of a new data point.

The next step is to define what it means for a grid to be dense. Since the total density over all grids is no larger than $\mu/(\mu - 1)$, it follows that the *average* density of each grid is no larger than $\mu/(\eta \cdot (\mu - 1))$. Therefore, a grid is defined as *dense* when its density is a constant times larger than this factor. This is essentially analogous to the concept of a c-micro-cluster defined in [31]. Analogous to the concept of an o-micro-cluster and a p-micro-cluster, the work in [25] divides the non-dense grid cells into *sparse grid cells* and *transitional grid cells*, with the use

of a smaller threshold on the density. Grid cells can change between the different states of being sparse, transitional, or dense, both because of the addition of new data points and because of decay.

One observation is that the number of grid cells $\eta = \prod_{i=1}^{d} p_i$ is exponentially dependent upon the dimensionality d. However, in practice, most of these grid cells are empty, and the information for empty grids need not be stored. This may not be sufficient in many real applications, where many outlier points may sporadically appear in the data. The work in [25] designs methods for identifying and removing such sporadic grids from the data. The maintained dense grids can then be consolidated into larger dense regions in the data. As in the case of [31], this is defined in the form of density-connected grids, where the adjacency of two dense grids is treated as a density connection. For the precise details of the dense region construction, we refer the reader to [25]. Thus, it is evident that grid-based and micro-cluster-based density clusters share a number of conceptual similarities at various stages of the algorithm.

One weakness of the approach is that a significant number of non-empty grid cells need to be discarded in order to keep the memory requirements in check. In many cases, such grid cells occur at the borders of the clusters. The discarding of such cells may lead to degradation in cluster quality. Therefore, a method has been proposed in [45] to design a variation of the *D-Stream* method (known as *DD-Stream*), which includes the data points at the borders into adjacent denser grids, which are retained by the algorithm. It has been shown that such an approach leads to some improvements in cluster quality. Methods based on shared density between micro-clusters are discussed in [39].

3.2.2 Other Grid-Based Algorithms

The method in [36] updates a full-dimensional grid of the incoming data stream. The clusters are discovered from the updated density values in this grid. At any given time, the density values in the grid can be used in order to determine the updated and most effective clusters.

An important point to be kept in mind is that grid-based algorithms require the discretization of the data along the different dimensions in order to create the grid cells. The granularity of the discretization is a critical design choice at the very beginning of the algorithm. Unfortunately, at the beginning of the stream arrival, very little may be known about how the underlying data points are distributed, and therefore it is difficult to choose the level of discretization along each dimension in an optimal way. Furthermore, the appropriate level of discretization for each data locality may be different, and a single global level of discretization may be suboptimal over different data localities.

Therefore, the work in [57] uses a dynamic approach called *Statsgrid* for the discretization process, wherein the data cells are recursively partitioned based on their local density. The algorithm starts off with cells of equal size. As data points are added to the cells and the number of points in a cell becomes sufficiently large,

the algorithm partitions the cell into two along one of the dimensions. This process can of course be repeated recursively each time any of the children cells become dense. At some point, the maximum level of allowed granularity is reached, and such a cell is called a *unit cell*, which cannot be further divided. We note that this approach naturally leads to a hierarchical clustering of the data, which can be very useful in many applications. Nevertheless, the work does not use temporal decay and therefore does not adjust very well to an evolving data stream.

This method has therefore been extended to the *CellTree* method [58], which allows decay in the statistics. It explicitly uses a *CellTree* data structure in order to maintain the hierarchical relationships among the grid cells. Furthermore, when the data cells decay with time, it may be possible to merge adjacent cells. Therefore, the method in [58] provides greater flexibility than discussed in the original algorithm.

4 Probabilistic Streaming Algorithms

One of the common methods for probabilistic clustering is that of *mixture modeling*, in which the data is assumed to be generated by a mixture of known distributions such as the Gaussian distribution. The parameters of this distribution are then typically learned with an EM algorithm from the actual data records [61]. The main argument in [61] is that probability density-based clustering algorithms are much more efficient than applying EM on the entire data set. On the other hand, it has been shown that it is possible to use the EM algorithm in order to provide an efficient update process for newly arriving data. Nevertheless, since the EM algorithm requires one to learn a large number of parameters, such an approach is unlikely to be effective when the underlying data is evolving rapidly.

Another area where probabilistic models are commonly used is for text clustering. A common technique that is used to create a soft clustering of the data for the case of text is that of *topic modeling* [43]. In this technique, soft clusters are associated with the data in which words and documents are probabilistically assigned to the different partitions. Since the topic modeling approach uses an EM algorithm, it can sometimes be slow in practice. Therefore, a method has been proposed in [20] for topic modeling over text streams. The work in [20] proposes online variants of three common batch algorithms for topic modeling. These correspond to the Latent Dirichlet Allocation (LDA) [22], Dirichlet Compound Multinomial (DCM) mixtures [30], and von-Mises Fisher (vMF) mixture models [21]. It was shown in [20] that the online variant of the vMF approach provides the best results. A detailed study of these topic modeling algorithms is beyond the scope of this survey. Interested readers are referred to [20].

5 Clustering High-Dimensional Streams

In many circumstances, data stream is very high-dimensional because a large number of features are available for the mining process. The high-dimensional case presents a special challenge to clustering algorithms even in the traditional domain of static data sets. This is because of the sparsity of the data in the high-dimensional case. In high-dimensional space, all pairs of points tend to be almost equidistant from one another. As a result, it is often unrealistic to define distance-based clusters in a meaningful way. Some recent work on high-dimensional data uses techniques for *projected clustering* which can determine clusters for a specific subset of dimensions [3, 17]. In these methods, the definitions of the clusters are such that each cluster is specific to a particular group of dimensions. This alleviates the sparsity problem in high-dimensional space to some extent. Even though a cluster may not be meaningfully defined on all the dimensions because of the sparsity of the data, some subset of the dimensions can always be found on which particular subsets of points form high-quality and meaningful clusters. Of course, these subsets of dimensions may vary over the different clusters. Such clusters are referred to as *projected clusters* [3].

5.1 The HPSTREAM Method

The micro-clustering method can also be extended to the case of high-dimensional projected stream clustering. The algorithm is referred to as *HPSTREAM*. In [15, 16], methods have been proposed for high-dimensional projected clustering of data streams. The basic idea is to use an (incremental) algorithm in which we associate a set of dimensions with each cluster. This corresponds to the standard micro-clustering method as discussed in [15], with the main difference that the distances from data points to clusters are computed on the basis of dimension-specific clusters. Therefore, additional information needs to be associated with a cluster in order to capture the set of dimensions associated with them. The set of dimensions is represented as a d-dimensional bit vector $\mathcal{B}(C_i)$ for each cluster structure in $\mathcal{FCS}$. This bit vector contains a 1 bit for each dimension which is included in cluster C_i. In addition, the maximum number of clusters k and the average cluster dimensionality l are used as an input parameter. The average cluster dimensionality l represents the average number of dimensions used in the cluster projection. An iterative approach is used in which the dimensions are used to update the clusters and vice versa. The structure in $\mathcal{FCS}$ uses a decay-based mechanism in order to adjust for evolution in the underlying data stream. For a data point, which arrived δt units ago, its weight is assumed to be $2^{-\lambda \cdot \delta t}$, where λ is the decay rate.

Therefore, the micro-clusters for the *HPSTREAM* algorithm contain decay-based statistics, wherein the micro-clusters are similar to the *CluStream* algorithm. Furthermore, the overall framework of the *HPSTREAM* algorithm is quite similar

because data points are assigned to micro-clusters on the basis of their projected distances. The main difference is that each component of the additive micro-cluster statistics is a decay-based weight. In addition, the bit vector corresponding to the choice of dimensions is stored with the micro-cluster statistics. Clearly, a number of changes need to be incorporated into the *CluStream* approach in order to account for these changes:

- We note that the decay-based statistics ensure that the weights of the micro-clusters change in each time stamp. However, it is not necessary to update the statistics at each time stamp, since all the micro-clusters decay at the same rate. Rather, we perform the update only when a new point is inserted into the data. When a new point is inserted and δx is the time interval since the last time, then all micro-cluster statistics are multiplied by $2^{-\lambda \cdot \delta x}$ before adding a data point to the micro-cluster statistics.
- The average distance to each cluster is now computed on the basis of the projected dimensions specific to that cluster. The bit vector in the micro-cluster statistics is used in order to decide on the exact set of dimensions to use.
- The projected dimensions in the different clusters are updated periodically, so that the most compact dimensions are retained for each cluster. The corresponding bit vector in the micro-cluster statistics is updated.

It has been shown in [15] that the incorporation of projections can significantly improve the effectiveness of the approach. The details are discussed in [15].

5.2 Other High-Dimensional Streaming Algorithms

A variety of other high-dimensional streaming clustering algorithms have been proposed after the first *HPSTREAM* framework. In particular, a grid-based algorithm was proposed in [51] for high-dimensional projected clustering of data streams. High-dimensional projected clustering has also been applied to other domains such as uncertain data. For example, methods for high-dimensional projected clustering of uncertain data streams have been proposed in [8, 62].

Most of the methods discussed in the literature use a k-means type approach, which fixes the number of clusters in the data. Furthermore, a k-means type approach also makes implicit assumptions about the *shapes* of the underlying clusters. The work in [54] proposes a density-based method for high-dimensional stream clustering. Such an approach has the virtue of recognizing the fact that the number and shape of the clusters in the stream may vary over time. The work in [54] proposes HDDSTREAM, which is a method for high-dimensional stream clustering. It generalizes the density-based approach proposed in [31] in order to incorporate subspace analysis in the clustering process. Since the method in [31] was originally designed to handle variations in the number of clusters in the stream, as well as different shapes of clusters, these virtues are inherited by the HDDSTREAM method of [54] as well.

A number of methods have also been proposed for high-dimensional projected clustering of dynamic data [49, 64]. While these methods can be effectively used for dynamic data, they do require access to the raw data for clustering purposes. Therefore, these methods are not streaming techniques in the strict sense.

6 Clustering Categorical Streams

Many data streams are defined on a domain of *discrete values* in which the attributes are unordered and take on one of a very large number of possible discrete values. In such cases, the cluster feature vector of the micro-clustering approach does not represent the values in the underlying data well. Therefore, methods are required in order to perform stream clustering algorithms in such scenarios. The simplest case of categorical stream clustering is that of *binary data* in which the data takes on values from $\{0, 1\}$. Binary data can be considered both quantitative and symbolic, and therefore almost all stream algorithms can be used directly for this case.

6.1 Clustering Binary Data Streams with k-Means

The simplest case of categorical data is binary data, in which the discrete attributes may take on only one of the two possible values. Binary data is also a special case of quantitative data because an ordering can always be assigned to discrete values as long as there are only two of them. Therefore, virtually all of the streaming algorithms can be used for binary data. Nevertheless, it can sometimes be useful to leverage a method which is specifically designed for the case of binary data.

An algorithm for utilizing optimizations of k-means algorithms for data streams is proposed in [56]. The main observation in [56] is that the binary transactions are often *sparse*. This can be used in order to greatly speed up the distance computations. Since distance computations form the bottleneck operation for such algorithms, a speedup of the distance computations also greatly speeds up the underlying algorithm. From a practical point of view, this means that only a small fraction of the features in the transaction take on the 1-value. Therefore, the general approach used in [56] is that first the distance of the null transaction to each of the centroids is computed. Subsequently, for each position in the transaction, the effect of that position on the distances is computed. Since many distance functions are separable functions across different dimensions, this can be achieved quite effectively. It has been shown in [56] that these speedups can be implemented very effectively at no reduction in quality of the underlying results.

6.2 The StreamCluCD Algorithm

The *Squeezer* algorithm was a one-pass algorithm for clustering categorical data [41]. The *StreamCluCD* approach [42] is the streaming extension of this framework. The essential idea behind the algorithm is that when a data point comes in, it is placed into a cluster of its own. For subsequent incoming points, we compute their similarity to the existing clusters in the data. If the incoming points are sufficiently similar to one of the existing clusters, then they are placed in that cluster. Otherwise, the incoming data point is placed in a cluster of its own. A key operation in the *StreamCluCD* algorithm is to maintain the frequency counts of the attribute values in each cluster. The lossy counting approach introduced in [52] is used for this purpose. The motivation for this is to reduce the memory footprint for maintaining the frequency counts. The issue of memory requirements becomes especially important when the number of possible discrete values of each attribute increases. This is referred to as the *massive domain* scenario and will be discussed in the next section.

6.3 Massive-Domain Clustering

Massive-domains are those data domains in which the number of possible values for one or more attributes is very large. Examples of such domains are as follows:

- In network applications, many attributes such as IP addresses are drawn over millions of possibilities. In a multidimensional application, this problem is further magnified because of the multiplication of possibilities over different attributes.
- Typical credit card transactions can be drawn from a universe of millions of different possibilities depending upon the nature of the transactions.
- Supermarket transactions are often drawn from a universe of millions of possibilities. In such cases, the determination of patterns that indicate different kinds of classification behavior may become infeasible from a space and computational efficiency perspective.

The massive-domain size of the underlying data restricts the *computational approach* which may be used for discriminatory analysis. Thus, this problem is significantly more difficult than the standard clustering problem in data streams. Space efficiency is a special concern in the case of data streams because it is desirable to hold most of the data structures in main memory in order to maximize the processing rate of the underlying data. Smaller space requirements ensure that it may be possible to hold most of the intermediate data in fast caches, which can further improve the efficiency of the approach. Furthermore, it may often be desirable to implement stream clustering algorithms in a wide variety of space-constrained architectures such as mobile devices, sensor hardware, or cell

processors. Such architectures present special challenges to the massive-domain case if the underlying algorithms are not space-efficient.

The problem of clustering can be extremely challenging from a space and time perspective in the massive-domain case. This is because one needs to retain the discriminatory characteristics of the most relevant clusters in the data. In the massive-domain case, this may entail storing the frequency statistics of a large number of possible attribute values. While this may be difficult to do explicitly, the problem is further intensified by the large volume of the data stream which prevents easy determination of the importance of different attribute values. The work in [4] proposes a sketch-based approach in order to keep track of the intermediate statistics of the underlying clusters. These statistics are used in order to make approximate determinations of the assignment of data points to clusters. A number of probabilistic results are provided in [4], which indicate that these approximations are sufficiently accurate to provide similar results to an infinite-space clustering algorithm with high probability.

The data stream $\mathcal{D}$ contains d-dimensional records denoted by $\overline{X_1} \ldots \overline{X_N} \ldots$. The attributes of record $\overline{X_i}$ are denoted by $(x_i^1 \ldots x_i^d)$. It is assumed that the attribute value x_i^k is drawn from the unordered domain set $J_k = \{v_1^k \ldots v_{M^k}^k\}$. The value of M^k denotes the domain size for the kth attribute. The value of M^k can be very large and may range in the order of millions or billions. From the point of view of a clustering application, this creates a number of challenges, since it is no longer possible to hold the cluster statistics in a space-limited scenario.

The work in [4] uses the count-min sketch [26] for the problem of clustering massive-domain data streams. In the count-min sketch, a hashing approach is utilized in order to keep track of the attribute value statistics in the underlying data. We use $w = \lceil \ln(1/\delta) \rceil$ pairwise independent hash functions, each of which maps onto uniformly random integers in the range $h = [0, e/\epsilon]$, where e is the base of the natural logarithm. The data structure itself consists of a two-dimensional array with $w \cdot h$ cells with a length of h and a width of w. Each hash function corresponds to one of w one-dimensional arrays with h cells each. In standard applications of the count-min sketch, the hash functions are used in order to update the counts of the different cells in this two-dimensional data structure. For example, consider a one-dimensional data stream with elements drawn from a massive set of domain values. When a new element of the data stream is received, we apply each of the w hash functions to map onto a number in $[0 \ldots h-1]$. The count of each of the set of w cells is incremented by 1. In order to *estimate* the count of an item, we determine the set of w cells to which each of the w hash functions maps and compute the minimum value among all these cells. Let c_t be the true value of the count being estimated. We note that the estimated count is at least equal to c_t, since we are dealing with nonnegative counts only, and there may be an over-estimation because of collisions among hash cells. As it turns out, a probabilistic upper bound to the estimate may also be determined. It has been shown in [26] that for a data stream with T arrivals, the estimate is at most $c_t + \epsilon \cdot T$ with probability at least $1 - \delta$.

```
Algorithm CSketch(Labeled Data Stream: D,
                  NumClusters: k)
begin
  Create k sketch tables of size w · h each;
  Initialize k sketch tables to null counts;
  repeat
    Receive next data point X from D;
    Compute the approximate dot product of incoming data
     point with each cluster-centroid with the use of the
     sketch tables;
    Pick the centroid with the largest approximate
      dot product to incoming point;
    Increment the sketch counts in the chosen table
     for all d dimensional value strings;
  until(all points in D have been processed);
end
```

Fig. 1 The sketch-based clustering algorithm (CSketch algorithm)

The *CSketch* algorithm uses the number of clusters k and the data stream $\mathcal{D}$ as input to the algorithm (Fig. 1). The clustering algorithm is partition-based and assigns incoming data points to the most similar cluster centroid. The frequency counts for the different attribute values in the cluster centroids are incremented with the use of the sketch table. These frequency counts can be maintained only approximately because of the massive-domain size of the underlying attributes in the data stream. Similarity is measured with the computation of the dot product function between the incoming point and the centroid of the different clusters. This computation can be performed only approximately in the massive-domain case, since the frequency counts for the values in the different dimensions cannot be maintained explicitly. For each cluster, we maintain the frequency sketches of the records which are assigned to it. Specifically, for each cluster, the algorithm maintains a separate sketch table containing the counts of the values in the incoming records. The same hash function is used for each table. The algorithm starts off by initializing the counts in each sketch table to 0. Subsequently, for each incoming record, we will update the counts of each of the cluster-specific hash tables. In order to determine the assignments of data points to clusters, the dot products of the d-dimensional incoming records to the frequency statistics of the values in the different clusters are computed. This is sometimes not possible to do explicitly, since the precise frequency statistics are not maintained. Let $q_r^j(x_i^r)$ represent the frequency of the value x_i^r in the jth cluster. Let m_i be the number of data points assigned to the jth cluster. Then, the d-dimensional statistics of the record $(x_i^1 \ldots x_i^d)$ for the jth cluster is given by $(q_1^j(x_i^1) \ldots q_d^j(x_i^d))$. Then, the frequency-based dot product $D^j(\overline{X_i})$ of the incoming record with statistics of cluster j is given by the dot product of the fractional frequencies $(q_1^j(x_i^1)/m_j \ldots q_d^j(x_i^d)/m_j)$ of the attribute values $(x_i^1 \ldots x_i^d)$ with the frequencies of these same attribute values within record $\overline{X_i}$. We note that the frequencies of the attribute values with the record $\overline{X_i}$ are

unit values corresponding to $(1, \ldots 1)$. Therefore, the corresponding dot product is the following:

$$D^j(\overline{X_i}) = \sum_{r=1}^{d} q_r^j(x_i^r)/m_j \tag{5}$$

The incoming record is assigned to the cluster for which the estimated dot product is the largest. We note that the value of $q_r^j(x_i^r)$ cannot be known exactly but can only be estimated approximately because of the massive-domain constraint. There are two key steps which use the sketch table during the clustering process:

- Updating the sketch table and other required statistics for the corresponding cluster for each incoming record.
- Comparing the similarity of the incoming record to the different clusters with the use of the corresponding sketch tables.

First, we discuss the process of updating the sketch table, once a particular cluster has been identified for assignment. For each record, the sketch table entries corresponding to the attribute values on the different dimensions are incremented. For each incoming record $\overline{X_i}$, the w hash functions are applied to the strings corresponding to the attribute values in it. Let m be the index of the cluster to which the data point is assigned. Then, exactly $d \cdot w$ entries in the sketch table for cluster m are updated by applying the w hash functions to each of the d strings which are denoted by $x_i^1 \oplus 1 \ldots x_i^d \oplus d$. The corresponding entries are incremented by one unit each.

In order to pick a cluster for assignment, the approximate dot products across different clusters need to be computed. The record is converted to its sketch representation by applying the w hash functions to each of these d different attribute values. We retrieve the corresponding d sketch table entries for each of the w hash functions and each cluster. For each of the w hash functions for the sketch table for cluster j, the d counts are simply estimates of the value of $q_1^j(x_i^1) \ldots q_d^j(x_i^d)$. Specifically, let the count for the entry picked by the lth hash function corresponding to the rth dimension of record $\overline{X_i}$ in the sketch table for cluster j be denoted by c_{ijlr}. Then, we estimate the dot product D_{ij} between the record $\overline{X_i}$ and the frequency statistics for cluster j as follows:

$$D_{ij} = \min_l \sum_{r=1}^{d} c_{ijlr}/m_j \tag{6}$$

The value of D_{ij} is computed over all clusters j, and the cluster with the largest dot product to the record $\overline{X_i}$ is picked for assignment.

It has been shown in [4] that this assignment process approximates an infinite-space clustering algorithm quite well. In addition, the experimental results in [4] show that the clustering process can be replicated almost exactly in practice with the use of this approximation process. Thus, the work in [4] proposes a fast and space-efficient method for clustering massive-domain data streams.

7 Text Stream Clustering

The problem of streaming text clustering is particularly challenging in the context of text data because of the fact that the clusters need to be continuously maintained in real time. One of the earliest methods for streaming text clustering was proposed in [67]. This technique is referred to as the *Online Spherical k-Means Algorithm (OSKM)*, which reflects the broad approach used by the methodology. This technique divides up the incoming stream into small segments, each of which can be processed effectively in main memory. A set of k-means iterations are applied to each such data segment in order to cluster them. The advantage of using a segment-wise approach for clustering is that since each segment can be held in main memory, we can process each data point multiple times as long as it is held in main memory. In addition, the centroids from the previous segment are used in the next iteration for clustering purposes. A decay factor is introduced in order to age out the old documents, so that the new documents are considered more important from a clustering perspective. This approach has been shown to be extremely effective in clustering massive text streams in [67].

A different method for clustering massive text and categorical data streams is discussed in [6]. The method discussed in [6] uses an approach which examines the relationship between outliers, emerging trends, and clusters in the underlying data. Old clusters may become inactive and eventually get replaced by new clusters. Similarly, when newly arriving data points do not naturally fit in any particular cluster, these need to be initially classified as outliers. However, as time progresses, these new points may create a distinctive pattern of activity which can be recognized as a new cluster. The temporal locality of the data stream is manifested by these new clusters. For example, the first web page belonging to a particular category in a crawl may be recognized as an outlier but may later form a cluster of documents of its own. On the other hand, the new outliers may not necessarily result in the formation of new clusters. Such outliers are true short-term abnormalities in the data since they do not result in the emergence of sustainable patterns. The approach discussed in [6] recognizes new clusters by first recognizing them as outliers. This approach works with the use of a summarization methodology, in which we use the concept of *condensed droplets* [6] in order to create concise representations of the underlying clusters.

As in the case of the OSKM algorithm, we ensure that recent data points are given greater importance than older data points. This is achieved by creating a time-sensitive weight for each data point. It is assumed that each data point has a time-dependent weight defined by the function $f(t)$. The function $f(t)$ is also referred to as the *fading function*. The fading function $f(t)$ is a non-monotonic decreasing function which decays uniformly with time t. The aim of defining a half-life is to quantify the rate of decay of the importance of each data point in the stream clustering process. The *decay rate* is defined as the inverse of the half-life of the data stream. We denote the decay rate by $\lambda = 1/t_0$. We denote the weight function of each point in the data stream by $f(t) = 2^{-\lambda \cdot t}$. From the perspective of the clustering

process, the weight of each data point is $f(t)$. It is easy to see that this decay function creates a half-life of $1/\lambda$. It is also evident that by changing the value of λ, it is possible to change the rate at which the importance of the historical information in the data stream decays.

When a cluster is created during the streaming process by a newly arriving data point, it is allowed to remain as a trend-setting outlier for at least one half-life. During that period, if at least one more data point arrives, then the cluster becomes an active and mature cluster. On the other hand, if no new points arrive during a half-life, then the trend-setting outlier is recognized as a true anomaly in the data stream. At this point, this anomaly is removed from the list of current clusters. We refer to the process of removal as *cluster death*. Thus, a new cluster containing one data point dies when the (weighted) number of points in the cluster is 0.5. The same criterion is used to define the death of mature clusters. A necessary condition for this criterion to be met is that the inactivity period in the cluster has exceeded the half-life $1/\lambda$. The greater the number of points in the cluster, the greater the level by which the inactivity period would need to exceed its half-life in order to meet the criterion. This is a natural solution since it is intuitively desirable to have stronger requirements (a longer inactivity period) for the death of a cluster containing a larger number of points.

The statistics of the data points are captured in summary statistics, which are referred to as *condensed droplets*. These represent the word distributions within a cluster and can be used in order to compute the similarity of an incoming data point to the cluster. The overall algorithm proceeds as follows. At the beginning of algorithmic execution, we start with an empty set of clusters. As new data points arrive, unit clusters containing individual data points are created. Once a maximum number k of such clusters have been created, we can begin the process of online cluster maintenance. Thus, we initially start off with a trivial set of k clusters. These clusters are updated over time with the arrival of new data points.

When a new data point $\overline{X}$ arrives, its similarity to each cluster droplet is computed. In the case of text data sets, the cosine similarity measure between $\overline{DF1}$ and $\overline{X}$ is used. The similarity value $S(\overline{X}, C_j)$ is computed from the incoming document $\overline{X}$ to every cluster C_j. The cluster with the maximum value of $S(\overline{X}, C_j)$ is chosen as the relevant cluster for data insertion. Let us assume that this cluster is C_{mindex}. We use a threshold denoted by $thresh$ in order to determine whether the incoming data point is an outlier. If the value of $S(\overline{X}, C_{mindex})$ is larger than the threshold $thresh$, then the point $\overline{X}$ is assigned to the cluster C_{mindex}. Otherwise, we check if some inactive cluster exists in the current set of cluster droplets. If no such inactive cluster exists, then the data point $\overline{X}$ is added to C_{mindex}. On the other hand, when an inactive cluster does exist, a new cluster is created containing the solitary data point $\overline{X}$. This newly created cluster replaces the inactive cluster. We note that this new cluster is a potential true outlier or the beginning of a new trend of data points. Further understanding of this new cluster may only be obtained with the progress of the data stream.

In the event that $\overline{X}$ is inserted into the cluster C_{mindex}, we update the statistics of the cluster in order to reflect the insertion of the data point and temporal

decay statistics. Otherwise, we replace the most inactive cluster by a new cluster containing the solitary data point $\overline{X}$. In particular, the replaced cluster is the least recently updated cluster among all inactive clusters. This process is continuously performed over the life of the data stream, as new documents arrive over time. The work in [6] also presents a variety of other applications of the stream clustering technique such as evolution and correlation analysis.

A different way of utilizing the temporal evolution of text documents in the clustering process is described in [40]. Specifically, the work in [40] uses *bursty features* as markers of new topic occurrences in the data stream. This is because the semantics of an up-and-coming topic are often reflected in the frequent presence of a few distinctive words in the text stream. At a given period in time, the nature of relevant topics could lead to bursts in specific features of the data stream. Clearly, such features are extremely important from a clustering perspective. Therefore, the method discussed in [40] uses a new representation, which is referred to as the *bursty feature representation* for mining text streams. In this representation, a time-varying weight is associated with the features depending upon its burstiness. This also reflects the varying importance of the feature to the clustering process. Thus, it is important to remember that a particular document representation is dependent upon the particular instant in time at which it is constructed.

Another issue that is handled effectively in this approach is an implicit reduction in dimensionality of the underlying collection. Text is inherently a high-dimensional data domain, and the pre-selection of some of the features on the basis of their burstiness can be a natural way to reduce the dimensionality of document representation. This can help in both the effectiveness and efficiency of the underlying algorithm.

The first step in the process is to identify the bursty features in the data stream. In order to achieve this goal, the approach uses Kleinberg's 2-state finite automaton model [47]. Once these features have been identified, the bursty features are associated with weights which depend upon their level of burstiness. Subsequently, a bursty feature representation is defined in order to reflect the underlying weight of the feature. Both the identification and the weight of the bursty feature are dependent upon its underlying frequency. A standard k-means approach is applied to the new representation in order to construct the clustering. It was shown in [40] that the approach of using burstiness improves the cluster quality. One criticism of the work in [40] is that it is mostly focused on the issue of improving effectiveness with the use of temporal characteristics of the data stream and does not address the issue of efficient clustering of the underlying data stream.

In general, it is evident that feature extraction is important for all clustering algorithms. While the work in [40] focuses on using temporal characteristics of the stream for feature extraction, the work in [50] focuses on using *phrase extraction* for effective feature selection. This work is also related to the concept of topic modeling, which will be discussed in detail in the next section. This is because the different topics in a collection can be related to the clusters in a collection. The work in [50] uses topic modeling techniques for clustering. The core idea in the work of [50] is that individual words are not very effective for a clustering algorithm because they miss the context in which the word is used. For example, the word "star" may either

refer to a celestial body or to an entertainer. On the other hand, when the phrase "fixed star" is used, it becomes evident that the word "star" refers to a celestial body. The phrases that are extracted from the collection are also referred to as *topic signatures*.

The use of such phrasal clarification for improving the quality of the clustering is referred to as *semantic smoothing* because it reduces the noise that is associated with semantic ambiguity. Therefore, a key part of the approach is to extract phrases from the underlying data stream. After phrase extraction, the training process determines a translation probability of the phrase to terms in the vocabulary. For example, the word "planet" may have high probability of association with the phrase "fixed star" because both refer to celestial bodies. Therefore, for a given document, a rational probability count may also be assigned to all terms. For each document, it is assumed that all terms in it are generated either by a topic signature model or by a background collection model.

The approach in [50] works by modeling the soft probability $p(w|C_j)$ for word w and cluster C_j. The probability $p(w|C_j)$ is modeled as a linear combination of two factors: (a) a maximum likelihood model that computes the probabilities of generating specific words for each cluster and (b) an indirect (translated) word membership probability that first determines the maximum likelihood probability for each topic signature, then multiplying with the conditional probability of each word, given the topic signature. We note that we can use $p(w|C_j)$ in order to estimate $p(d|C_j)$ by using the product of the constituent words in the document. For this purpose, we use the frequency $f(w, d)$ of word w in document d.

$$p(d|C_j) = \prod_{w \in d} p(w|C_j)^{f(w,d)} \tag{7}$$

We note that in the static case, it is also possible to add a background model in order to improve the robustness of the estimation process. This is however not possible in a data stream because of the fact that the background collection model may require multiple passes in order to build effectively. The work in [50] maintains these probabilities in online fashion with the use of a *cluster profile*, which weights the probabilities with the use of a fading function. We note that the concept of cluster profile is analogous to the concept of condensed droplet introduced in [6]. The key algorithm (denoted by OCTS) is to maintain a dynamic set of clusters into which documents are progressively assigned with the use of similarity computations. It has been shown in [50] how the cluster profile can be used in order to efficiently compute $p(d|C_j)$ for each incoming document. This value is then used in order to determine the similarity of the documents to the different clusters. This is used in order to assign the documents to their closest cluster. We note that the methods in [6, 50] share a number of similarities in terms of (a) maintenance of cluster profiles, (b) use of cluster profiles (or condensed droplets) to compute similarity and assignment of documents to most similar clusters, and (c) the rules used to decide when a new singleton cluster should be created or one of the older clusters should be replaced.

The main difference between the two algorithms is the technique which is used in order to compute cluster similarity. The OCTS algorithm uses the probabilistic computation $p(d|C_j)$ to compute cluster similarity, which takes the phrasal information into account during the computation process. One observation about OCTS is that it may allow for very similar clusters to co-exist in the current set. This reduces the space available for distinct cluster profiles. A second algorithm called OCTSM is also proposed in [50], which allows for merging of very similar clusters. Before each assignment, it checks whether pairs of similar clusters can be merged on the basis of similarity. If this is the case, then we allow the merging of the similar clusters and their corresponding cluster profiles. Detailed experimental results on the different clustering algorithms and their effectiveness are presented in [50]. A comprehensive review of text stream clustering algorithms may be found in the chapters on data clustering and streaming algorithms in [2].

8 Other Scenarios for Stream Clustering

Data stream clustering is a fundamental problem, and numerous other domains arise, in which the data occurs naturally in the streaming context. In this section, we provide a brief introduction to these different data domains, along with pointers to the relevant literature.

8.1 Clustering Uncertain Data Streams

Uncertain data streams arise quite often in the context of sensor data or other hardware collection technology such as RFID in which there are significant errors in the data collection process. In many of these cases, the errors in the data can be approximated either in terms of statistical parameters such as the standard deviation or in terms of probability density functions (pdfs). Such statistical information increases the richness of the noisy data because it provides information about the parts of data which are less reliable and should therefore be emphasized less in the mining process.

In this context, a method called *UMicro* for clustering uncertain data streams was proposed in [7]. This method enhances the micro-clusters with additional information about the uncertainty of the data points in the clusters. This information is used in order to improve the quality of the distance functions for the assignments. This approach was further improved for the case of projected clustering of uncertain data streams [8, 62]. We are not providing a detailed discussion of these methods, since they are discussed in detail in the chapter on uncertain data clustering.

8.2 Clustering Graph Streams

Graph streams are created by edge-centered activity in numerous social and information networks. Many different kinds of streaming and clustering models are possible, depending upon the application scenario. These different models are as follows:

- The stream is a sequence of *objects*, each of which is a small graph containing a subset of nodes and edges. We wish to determine similar *objects* in the stream based on the similarities between the nodes and edges. For example, the DBLP bibliographic network has an incoming stream of graph objects corresponding to the co-authored papers. It may be desirable to determine clusters of objects with similar structure. An algorithm for this problem, known as $Gmicro$, was proposed in [9]. The micro-clustering representation is extended to handle edges, and a sketch-based compression is used on the edges in order to reduce the impact of the massive-domain of the edges.
- The stream is a sequence of *objects*, each of which is a small graph containing a subset of nodes and edges. We wish to determine node sets which co-occur frequently in these objects and are also densely connected together. A method was proposed in [11] with the use of min-hash-based graph stream summarization.
- The stream is a sequence of either *objects* or *edges*. It is desirable to determine dense node clusters in the graph.

The last problem is particularly general and is also related to general problem of dynamic community detection. In this context, a method was proposed for creating dynamic partitions of graphs with the use of structural reservoir sampling of edge streams [10]. While the work in [10] is targeted to outlier detection, a key intermediate step in the process is the dynamic generation of node clusters from the edge stream. The work in [29] has further refined the structural reservoir sampling techniques of [10] in order to provide more effective methods for node clustering in graph streams.

In many cases, such as social networks, content information is associated with the structure in the network. For example, the tweets in a *Twitter* stream have both structure and content. Such streams are referred to as *social streams*. The clustering of such streams requires the use of both structural information and content in the process. The work in [12] has designed methods for clustering social streams. Sketch-based methods are used in order to summarize the content in the social stream and use it for the clustering process. In addition, it has been shown in [12] how this approach may be used for event detection.

In the conventional streaming model, it is assumed that only one pass is allowed over the data, and the amount of memory available to the application is constant, irrespective of stream length. A technique for determining the densest subgraph has been proposed in [19] in a *weakly streaming model*, where a limited number of passes (more than one) are allowed over the data, and the amount of available

memory is sub-linear in the size of the input. This approach is able to determine the densest subgraph in passes which grow logarithmically with the graph size.

8.3 Distributed Clustering of Data Streams

In the context of sensor networks, the stream data is often available only in a *distributed setting*, in which large volumes of data are collected separately at the different sensors. A natural approach for clustering such data is to transmit all of the data to a centralized server. The clustering can then be performed at the centralized server in order to determine the final results. Unfortunately, such an approach is extremely expensive in terms of its communication costs because of the large volume of the data which must be transmitted to the centralized server. Therefore, it is important to design a method that can reduce the communication costs among the different processors. A method proposed in [27] performs local clustering at each node and merges these different clusters into a single global clustering at low communication cost. Two different methods are proposed in this work. The first method determines the cluster centers by using a *furthest point algorithm*, on the current set of data points at the local site. In the furthest point algorithm, the center of a cluster is picked as the furthest point to the current set of centers. For any incoming data point, it is assigned to its closest center, as long the distance is within a certain factor of an optimally computed radius. Otherwise, a re-clustering is forced by applying the furthest point algorithm on current set of points. After the application of the furthest point algorithm, the centers are transmitted to the central server, which then computes a global clustering from these local centers over the different nodes. These global centers can then be transmitted to the local nodes if desired. One attractive feature of the method is that an approximation bound is proposed on the quality of the clustering. A second method for distributed clustering proposed in [27] is the *parallel guessing algorithm*.

A variety of other methods have also been proposed for distributed clustering of data streams. For example, techniques for distributed data stream clustering with the use of the k-medians approach were proposed in [63]. Another method for distributed sensor stream clustering which reduces the dimensionality and communication cost by maintaining an online discretization may be found in [59]. Finally, a method for distributed clustering of data streams with the use of the EM approach is proposed in [69].

9 Discussion and Conclusions

The problem of clustering is fundamental to a wide variety of streaming applications because of its natural applicability to summarizing and representing the data in very small space. A wide variety of techniques have been proposed in the literature for

stream clustering, such as partitioning methods, density-based methods, probabilistic methods, etc. The stream clustering method has also been extended to other data domains such as categorical, text, and uncertain data. Finally, methods have also been proposed for clustering graph streams.

Many further directions of research are likely for this problem. These are as follows:

- Heterogeneous data streams are becoming extremely common because of applications such as social networks, in which large amounts of heterogeneous content are posted over time. It is desirable to determine clusters among these groups of heterogeneous objects.
- It is desirable to use a combination of links and content in order to determine clusters from heterogeneous activity streams. This direction of research has also found increasing interest in the community. This is a further layer of complexity over the graph streaming scenario.

Furthermore, it would also be interesting to perform clustering streams from multiple sources, while simultaneously learning the nature of the dynamic relationships between the different streams.

References

1. C. Aggarwal. *Data Streams: Models and Algorithms*, Springer, 2007.
2. C. Aggarwal, and C. X. Zhai, *Mining Text Data*, Springer, 2012.
3. C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, and J.-S. Park. Fast algorithms for projected clustering. In *ACM SIGMOD Conference*, 1999.
4. C. Aggarwal. A Framework for Clustering Massive-Domain Data Streams. In *ICDE Conference*, 2009.
5. C. Aggarwal, and P. Yu. Finding Generalized Projected Clusters in High Dimensional Spaces. In *ACM SIGMOD Conference*, 2000.
6. C. Aggarwal, and P. Yu. A Framework for Clustering Massive Text and Categorical Data Streams. In *SIAM Conference on Data Mining*, 2006.
7. C. Aggarwal, and P. Yu. A Framework for Clustering Uncertain Data Streams, *IEEE ICDE Conference*, 2008.
8. C. Aggarwal. On High-Dimensional Projected Clustering of Uncertain Data Streams, *IEEE ICDE Conference*, 2009.
9. C. Aggarwal, and P. Yu. On Clustering Graph Streams, *SDM Conference*, 2010.
10. C. Aggarwal, Y. Zhao, and P. Yu. Outlier Detection in Graph Streams, *ICDE Conference*, 2011.
11. C. Aggarwal, Y. Li, P. Yu, and R. Jin. On Dense Pattern Mining in Graph Streams, *VLDB Conference*, 2010.
12. C. Aggarwal, and K. Subbian. Event Detection in Social Streams, *SIAM Conference on Data Mining*, 2012.
13. C. Aggarwal. On Change Diagnosis in Evolving Data Streams. In *IEEE TKDE*, 17(5), 2005.
14. C. Aggarwal, J. Han, J. Wang, and P. Yu. A Framework for Clustering Evolving Data Streams. In *VLDB Conference*, 2003.
15. C. Aggarwal, J. Han, J. Wang, and P. Yu. A Framework for Projected Clustering of High Dimensional Data Streams. In *VLDB Conference*, 2004.

16. C. Aggarwal, J. Han, J. Wang, and P. Yu. On High Dimensional Projected Clustering of Data Streams, *Data Mining and Knowledge Discovery Journal*, 10(3), pp. 251–273, 2005.
17. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *ACM SIGMOD Conference*, 1998.
18. M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering Points To Identify the Clustering Structure. In *ACM SIGMOD Conference*, 1999.
19. B. Bahamani, R. Kumar, V. Vassilvitskii. Densest Subgraph Mining in Streaming and MapReduce, *VLDB Conference*, 2012.
20. A. Banerjee, and S. Basu. Topic Models over Text Streams: A Study of Batch and Online Unsupervised Learning, *SIAM Conference*, 2007.
21. A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von Mises-Fisher distributions. In *Journal of Machine Learning Research*, 6: pages 1345–1382, 2005.
22. D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation, *Journal of Machine Learning Research*, 3: pp. 993–1022, 2003.
23. P. Bradley, U. Fayyad, and C. Reina. Scaling Clustering Algorithms to Large Databases. In *ACM KDD Conference*, 1998.
24. F. Cao, M. Ester, W. Qian, A. Zhou. Density-based Clustering over an Evolving Data Stream with Noise, In *SDM Conference*, 2006.
25. Y. Chen, and L. Tu. Density-based clustering for real time stream data, *ACM KDD Conference*, 2007.
26. G. Cormode and S. Muthukrishnan. An Improved Data-Stream Summary: The Count-min Sketch and its Applications. In *Journal of Algorithms*, 55(1), 2005.
27. G. Cormode, S. Muthukrishnan, and W. Zhuang. Conquering the divide: Continuous clustering of distributed data streams. *ICDE Conference*, 2007.
28. J. de Andrade Silva, E. Hruschka, and J. Gama. An evolutionary algorithm for clustering data streams with a variable number of clusters. *Expert Systems with Applications*, 67, pp. 228–238, 2017.
29. A. Eldawy, R. Khandekar, and K. L. Wu. On clustering Streaming Graphs, *ICDCS Conference*, 2012.
30. C. Elkan. Clustering documents with an exponential family approximation of the Dirichlet compound multinomial distribution. *ICML Conference*, 2006.
31. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *ACM KDD Conference*, 1996.
32. F. Farnstrom, J. Lewis, and C. Elkan. Scalability for Clustering Algorithms Revisited. *ACM SIGKDD Explorations*, 2(1):pp. 51–57, 2000.
33. D. Fisher. Knowledge Acquisition via incremental conceptual clustering. *Machine Learning*, 2: pp. 139–172, 1987.
34. M. Franz, T. Ward, J. McCarley, and W.-J. Zhu. Unsupervised and supervised clustering for topic tracking. *ACM SIGIR Conference*, 2001.
35. G. P. C. Fung, J. X. Yu, P. Yu, and H. Lu. Parameter Free Bursty Events Detection in Text Streams, *VLDB Conference*, 2005.
36. J. Gao, J. Li, Z. Zhang, and P.-N. Tan. An incremental data stream clustering algorithm based on dense units detection. In *PAKDD Conference*, 2005.
37. S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams. In *IEEE FOCS Conference*, 2000.
38. S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In *ACM SIGMOD Conference*, 1998.
39. M. Hahsler and M. Bolanos. Clustering data streams based on shared density between micro-clusters. *IEEE Transactions on Knowledge and Data Engineering*, 28(6), pp. 1449–1461, 2016.
40. Q. He, K. Chang, E.-P. Lim, and J. Zhang. Bursty feature representation for clustering text streams. *SDM Conference*, 2007.

41. Z. He, X. Xu, and S. Deng. Squeezer: An Efficient Algorithm for Clustering Categorical Data, *Journal of Computer Science and Technology (JCST)*, 17(5), pp. 611–624, 2002.
42. Z. He, X. Xu, S. Deng, and J. Huang. Clustering Categorical Data Streams, *Published Online on Arxiv*, December 2004. http://arxiv.org/ftp/cs/papers/0412/0412058.pdf
43. T. Hoffman. Probabilistic Latent Semantic Indexing, *ACM SIGIR Conference*, 1999.
44. A. Jain, R. Dubes. Algorithms for Clustering Data, *Prentice Hall,* New Jersey, 1998.
45. C. Jia, C. Tan, and A. Yong. A Grid and Density-based Clustering Algorithm for Processing Data Stream, *International Conference on Genetic and Evolutionary Computing*, 2008.
46. L. Kaufman, and P. Rousseuw. *Finding Groups in Data- An Introduction to Cluster Analysis*. Wiley Series in Probability and Math. Sciences, 1990.
47. J. Kleinberg, Bursty and hierarchical structure in streams, *ACM KDD Conference*, pp. 91–101, 2002.
48. P. Kranen, I. Assent, C. Baldauf, and T. Seidl. The ClusTree: Indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, 29(2), pp. 249–272, 2011. http://link.springer.com/article/10.1007%2Fs10115-010-0342-8
49. H.-P. Kriegel, P. Kroger, I. Ntoutsi, and A. Zimek. Density based subspace clustering over dynamic data. *SSDBM Conference*, 2011.
50. Y.-B. Liu, J.-R. Cai, J. Yin, and A. W.-C. Fu. Clustering Text Data Streams, *Journal of Computer Science and Technology*, Vol. 23(1), pp. 112–128, 2008.
51. Y. Lu, Y. Sun, G. Xu, and G. Liu. A Grid-based Clustering Approach for High-Dimensional Data Streams, *Advanced Data Mining and Applications*, 2005.
52. G. Manku, and R. Motwani. Approximate Frequency Counts over Data Streams, *VLDB Conference*, 2002.
53. R. Ng, and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Very Large Data Bases Conference*, 1994.
54. I. Ntoutsi, A. Zimek, T. Palpanas, H.-P. Kriegel, and A. Zimek. Density-based Projected Clustering over High Dimensional Data Streams, In *SDM Conference*, 2012.
55. L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-Data Algorithms For High-Quality Clustering. In *ICDE Conference*, 2002.
56. C. Ordonez. Clustering Binary Data Streams with K-means. In *Data Mining and Knowledge Discovery Workshop*, 2003.
57. N. H. Park, and W. S. Lee. Statistical Grid-based Clustering over Data Streams, *ACN SIGMOD Record*, 33(1), March, 2004. http://www09.sigmod.org/sigmod/record/issues/0403/A15.park-lee.pdf
58. N. H. Park, and W. S. Lee. Cell trees: An adaptive synopsis structure for clustering multi-dimensional on-line data streams. *Data and Knowledge Engineering*, 63(2), pp. 528–549, 2007.
59. P. Rodrigues, J. Gama, and L. Lopes. Clustering Distributed Sensor Data Streams, *PKDD Conference*, 2008.
60. B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.
61. M. Song, and H. Wang. Highly efficient incremental estimation of Gaussian Mixture Models for online data stream clustering, *SPIE*, 2005.
62. C. Zhang, M. Gao, and A. Zhou. Tracking High Quality Clusters over Uncertain Data Streams, *ICDE Conference*, 2009.
63. Q. Zhang, J. Liu, and W. Wang. Approximate clustering of distributed data streams, *ICDE Conference*, 2008.
64. Q. Zhang, J. Liu, and W. Wang. Incremental subspace clustering over multiple data streams. *ICDM Conference*, 2007.
65. T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *ACM SIGMOD Conference*, 1996.
66. Y. Zhao, S. Liang, Z. Ren, J. Ma, E. Yilmaz, and M. de Rijke. Explainable user clustering in short text streams. *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 155–164, 2016.

67. S. Zhong. Efficient Streaming Text Clustering. *Neural Networks*, Volume 18, Issue 5–6, 2005.
68. A. Zhou, F. Cao, W. Qian, and C. Jin. Tracking Clusters in Evolving Data Streams over Sliding Windows, *Knowledge and Information Systems*, 15(2), pp. 181–214, 2008.
69. A. Zhou, F. Cao, Y. Yan, C. Sha, and X. He. Distributed Clustering of Data Streams: A Fast EM-based Approach, *ICDE Conference*, 2007.

Introduction to Deep Learning

Lihi Shiloh-Perl and Raja Giryes

1 General Overview

Neural networks (NNs) have revolutionized the modern day-to-day life. Their significant impact is present even in our most basic actions, such as ordering products online via Amazon's Alexa or passing the time with online video games against computer agents. The NN effect is evident in many more occasions, for example, in medical imaging, NNs are utilized for lesion detection and segmentation [55, 5], and tasks such as text-to-speech [53, 171] and text-to-image [145] have remarkable improvements thanks to this technology. In addition, the advancements they have caused in fields such as natural language processing (NLP) [35, 206, 114], optics [158, 57], image processing [154, 204], and computer vision (CV) [13, 48] are astonishing, creating a leap forward in technology such as autonomous driving [20, 117], face recognition [153, 192, 33], anomaly detection [97], text understanding [80], and art [50, 79], to name a few. Its influence is powerful and is continuing to grow.

The NN journey began in the mid-1960s with the publication of the perceptron [149]. Its development was motivated by the formulation of the human neuron activity [118] and research regarding the human visual perception [73]. However, quite quickly, a deceleration in the field was experienced, which lasted for almost three decades. This was mainly the result of lack of theory with respect to the possibility of training the (single-layer) perceptron and a series of theoretical results that emphasized its limitations, where the most remarkable one is its inability to learn the XOR function [120].

L. Shiloh-Perl · R. Giryes (✉)
School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel
e-mail: lihishiloh@mail.tau.ac.il; raja@tauex.tau.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_14

This *NN ice age* came to a halt in the mid-1980s, mainly with the introduction of the multi-layer perceptron (MLP) and the backpropagation algorithm [151]. Furthermore, the revolutionary convolutional layer was presented [101], where one of its notable achievements was successfully recognizing hand-written digits [100].

While some other significant developments have happened in the following decade, such as the development of the long-short-term memory (LSTM) machine [68], the field experienced another deceleration. Questions were arising with no adequate answers especially with respect to the non-convex nature of the used optimization objectives, overfitting the training data, and the challenge of vanishing gradients. These difficulties led to a second *NN winter*, which lasted two decades. In the meantime, classical machine learning techniques were developed and attracted much academic and industry attention. One of the prominent algorithms was the newly proposed Support Vector Machine (SVM) [25], which defined a convex optimization problem with a clear mathematical interpretation [24]. These properties increased its popularity and usage in various applications.

The twenty-first century began with some advancements in neural networks in the areas of speech processing and Natural Language Processing (NLP). Hinton et al. [66] proposed a method for layer-wise initial training of neural networks, which leveraged some of the challenges in training networks with several layers. However, the great NN *tsunami* truly hit the field with the publication of *AlexNet* in 2012 [93]. In this paper, Krizhevsky et al. presented a neural network that achieved state-of-the-art performance on the ImageNet [32] challenge, where the goal is to classify images into 1000 categories using 1.2 Million images for training and 150,000 images for testing. The improvement over the runner-up, which relied on hand-crafted features and one of the best classification techniques of that time, was notable—more than 10%. This caused the whole research community to understand that neural networks are way more powerful than what was thought and they bear a great potential for many applications. This led to a myriad of research works that applied NNs for various fields showing their great advantage. In Table 2 (provided at the end of the chapter), we present a (very partial) list of different tasks and some selected neural networks that address them.

Nowadays, it is safe to say that almost every research field has been affected by this NN *tsunami* wave, experiencing significant improvements in abilities and performance. Many of the tools used today are very similar to the ones used in the previous phase of NN. Indeed, some new regularization techniques such as batch normalization [75] and dropout [172] have been proposed. Yet, the key enablers for the current success are the large amounts of data available today that are essential for large NN training and the developments in GPU computations that accelerate the training time significantly (sometimes even leading to $\times 100$ speed-up compared to training on a conventional CPU). The advantages of NN are remarkable especially at large scales. Thus, having large amounts of data and the appropriate hardware to process them is vital for their success.

A major example of a tool that did not exist before is the Generative Adversarial Network (GAN) [54]. In 2014, Goodfellow et al. published this novel framework for learning data distribution. The framework is composed of two

Fig. 1 Class-conditional samples generated by a GAN [8]

models, a generator and a discriminator, trained as adversaries. The generator is trained to capture the data distribution, while the discriminator is trained to differentiate between generated ("fake") data and real data. The goal is to let the generator synthesize data, which the discriminator fails to discriminate from the real one. The GAN architecture has been used in more and more applications since its introduction in 2014. One such application is the rendering of real scene images where GANs have been very successful [51, 215]. For example, Brock et al. introduced the BigGAN [8] architecture that exhibited impressive results in creating high-resolution images, shown in Fig. 1. While most GAN techniques learn from a set of images, recently it has been successfully demonstrated that one may even train a GAN just using one image [156]. Other GAN applications include inpainting [109, 208], retargeting [159], 3D modeling [1], semi-supervised learning [43], domain adaptation [69], and more.

While neural networks are very successful, the theoretical understanding behind them is still missing. In this respect, there are research efforts that try to provide a mathematical formulation that explains various aspects of NN [189]. For example, they study NN properties such as their optimization [175], generalization [78], and expressive power [152, 128].

The rest of the chapter is organized as follows. In Sect. 2, the basic structure of a NN is described, followed by details regarding popular loss functions and metric learning techniques used today (Sect. 3). We continue with an introduction to the NN training process in Sect. 4, including a mathematical derivation of backpropagation and training considerations. Section 5 elaborates on the different optimizers used during training, after which Sect. 6 presents a review of common regularization schemes. Section 7 details advanced NN architecture with state-of-the-art performances, and Sect. 8 concludes the chapter by highlighting some current important NN challenges.

2 Basic NN Structure

The basic building block of an NN consists of a linear operation followed by a nonlinear function. Each building block consists of a set of parameters, termed weights, and biases (sometimes the term weights includes also the biases), which

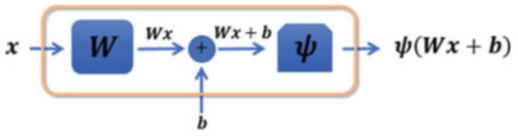


Fig. 2 NN building block consists of a linear and a nonlinear elements. The weights **W** and biases **b** are the parameters of the layer

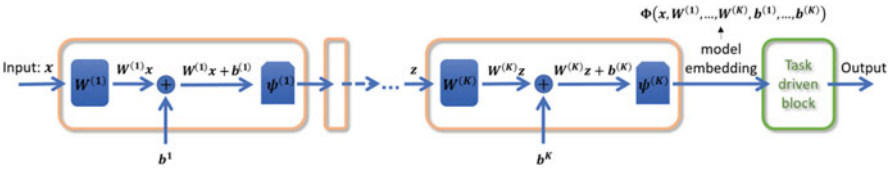


Fig. 3 NN layered structure: concatenation of N building blocks, e.g., model layers

are updated in the training process with the goal of minimizing a predefined loss function.

Assume an input data $\mathbf{x} \in \mathbb{R}^{d_0}$, and the output of the building block is of the form $\psi(\mathbf{W}\mathbf{x}+\mathbf{b})$, where $\psi(\cdot)$ is a nonlinear function, $\mathbf{W} \in \mathbb{R}^{d_1 \times d_0}$ is the linear operation, and $\mathbf{b} \in \mathbb{R}^{d_1}$ is the bias. See Fig. 2 for an illustration of a single building block.

To form an NN model, such building blocks are concatenated one to another in a layered structure that allows the input data to be gradually processed as it propagates through the network. Such a process is termed the (feed-)forward pass. Following it, during training, a backpropagation process is used to update the NN parameters, as elaborated in Sect. 4.1. In inference time, only the forward pass is used.

Figure 3 illustrates the concatenation of K building blocks, e.g., layers. The intermediate output at the end of the model (before the "task-driven block") is termed the *network embedding*, and it is formulated as follows:

$$\Phi(\mathbf{x}, \mathbf{W}^{(1)}, ..., \mathbf{W}^{(K)}, \mathbf{b}^{(1)}, ..., \mathbf{b}^{(K)}) = \psi(\mathbf{W}^{(K)}...\psi(\mathbf{W}^{(2)}\psi(\mathbf{W}^{(1)}\mathbf{x}+\mathbf{b}^{(1)}) + \mathbf{b}^{(2)})... + \mathbf{b}^{(K)}). \tag{1}$$

The final output (prediction) of the network is estimated from the network embedding of the input data using an additional task-driven layer. A popular example is the case of classifications, where this block is usually a linear operation followed by the *cross-entropy* loss function (detailed in Sect. 3).

When approaching the analysis of data with varying length, such as sequential data, a variant of the aforementioned approach is used. A very popular example for such a neural network structure is the recurrent neural network (RNN) [77]. In a vanilla RNN model, the network receives at each time-step just a single input but with a feedback loop calculated using the result of the same network in the previous time-step (see an illustration in Fig. 4). This enables the network to "remember" information and support multiple inputs and produce one or more outputs.

More complex RNN structures include performing bidirectional calculations or adding gating to the feedback and the input received by the network. The most known complex RNN architecture is the long-short-term memory (LSTM) [68, 52],

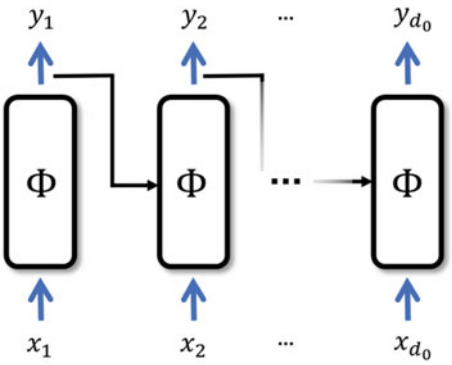


Fig. 4 Recurrent NN (RNN) illustration for time series data. The feedback loop introduces time-dependent characteristics to the NN model using an element-wise function. The weights are the same along all time-steps

which adds gates to the RNN. These gates decide what information from the current input and the past will be used to calculate the output and the next feedback, as well as what information to mask (i.e., causing the network to forget). This enables an easier combination of past and present information. It is commonly used for time series data in domains such as NLP and speech processing.

Another common network structure is the *encoder–decoder* architecture. The first part of the model, the encoder, reduces the dimensions of the input to a compact feature vector. This vector functions as the input to the second part of the model, the decoder. The decoder increases its dimension, usually, back to the original input size. This architecture essentially learns to compress (encode) the input to an efficiently small vector and then decode the information from its compact representation. In the context of regular feed-forward NN, this model is known as autoencoder [170] and is used for several tasks such as image denoising [146], image captioning [191], feature extraction [190], and segmentation [2]. In the context of sequential data, it is used for tasks such as translation, where the decoder generates a translated sentence from a vector representing the input sentence [177, 21].

2.1 Common Linear Layers

A common basic NN building block is the fully connected (FC) layer. A network composed of a concatenation of such layers is termed Multi-Layer Perceptron (MLP) [150]. The FC layer connects every neuron in one layer to every neuron in the following layer, i.e., the matrix $\mathbf{W}$ is dense. It enables information propagation from all neurons to all the ones following them. However, it may not maintain spatial information. Figure 5 illustrates a network with FC layers.

The convolutional layer [99, 101] is another very common layer. We discuss here the 2D case, where the extension to other dimension is straightforward. This layer applies one or multiple convolution filters to its input with kernels of size $W \times H$. The output of the convolution layer is commonly termed a *feature map*.

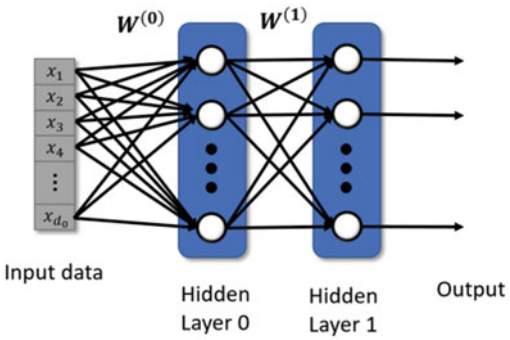


Fig. 5 Fully connected layers

Each neuron in a feature map receives inputs from a set of neurons from the previous layer, located in a small neighborhood defined by the kernel size. If we apply this relationship recursively, we can find the part of the input that affects each neuron at a given layer, i.e., the area of visible context that each neuron sees from the input. The size of this part is called the *receptive field*. It impacts the type and size of visual features each convolution layer may extract, such as edges, corners, and even patterns. Since convolution operations maintain spatial information and are translation equivariant, they are very useful, namely, in image processing and CV.

If the input to a convolution layer has some arbitrary third dimension, for example, 3 channels in an RGB image ($C = 3$) or some $C > 1$ channels from an output of a hidden layer in the model, the kernel of the matching convolution layer should be of size $W \times H \times C$. This corresponds to applying a different convolution for each input channel separately and then summing the outputs to create one feature map. The convolution layer may create a multi-channel feature map by applying multiple filters to the input, i.e., using a kernel of size $W \times H \times C_{\text{in}} \times C_{\text{out}}$, where C_{in} and C_{out} are the number of channels at the input and output of the layer, respectively.

2.2 Common Nonlinear Functions

The nonlinear functions defined for each layer are of great interest since they introduce the nonlinear property to the model and can limit the propagating gradient from vanishing or exploding (see Sect. 4).

Nonlinear functions that are applied element-wise are known as *activation functions*. Common activation functions are the Rectified Linear Unit (ReLU) [28], leaky ReLU [202], Exponential Linear Unit (ELU) [22], hyperbolic tangent (tanh), and sigmoid. There is no universal rule for choosing a specific activation function, and however, ReLUs and ELUs are currently more popular for image processing and CV, while sigmoid and tanh are more common in speech and NLP. Figure 6 presents

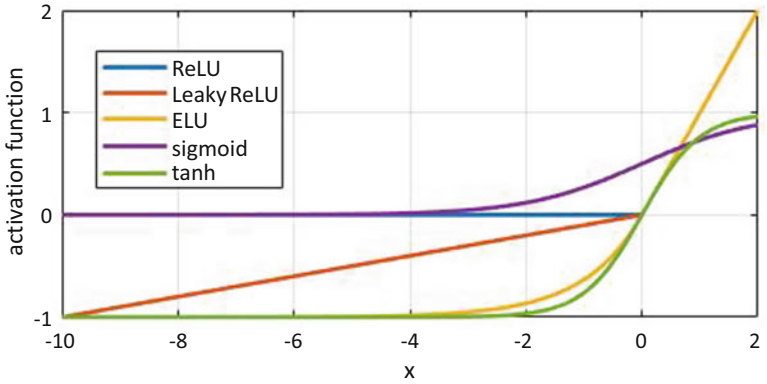


Fig. 6 Different activation functions. Leaky ReLU with $\alpha = 0.1$ and ELU with $\alpha = 1$

Table 1 Mathematical expressions for nonlinear activation functions

Function	Formulation $s(x)$	Derivative $\frac{ds(x)}{dx}$	Function output range
ReLU	$\begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$	$\begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$
Leaky ReLU	$\begin{cases} \alpha x, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$	$\begin{cases} \alpha, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
ELU	$\begin{cases} \alpha(e^x - 1), & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$	$\begin{cases} \alpha e^x, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases}$	$[-\alpha, \infty)$
Sigmoid	$\frac{1}{1+e^{-x}}$	$\frac{e^{-x}}{(1+e^{-x})^2}$	$(0, 1)$
tanh	$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$	$1 - \tanh^2(x)$	$(-1, 1)$

the response of the different activation functions and Table 1 their mathematical formulation.

Other common nonlinear operations in an NN model are the *pooling* functions. They are aggregation operations that reduce dimensionality while keeping dominant features. Assume a pooling size of q and an input vector to a hidden layer of size d, $\mathbf{z} = [z_1, z_1, ..., z_d]$. For every $m \in [1, d]$, the subset of the input vector $\tilde{\mathbf{z}} = [z_m, z_{m+1}, ..., z_{q+m}]$ may undergo one of the following popular pooling operations:

1. Max pooling: $g(\tilde{\mathbf{z}}) = \max_i \tilde{\mathbf{z}}$
2. Mean pooling: $g(\tilde{\mathbf{z}}) = \frac{1}{q} \sum_{i=m}^{q+m} z_i$
3. ℓ_p pooling: $g(\tilde{\mathbf{z}}) = \sqrt[p]{\sum_{i=m}^{q+m} z_i^p}$

All pooling operations are characterized by a stride, s, that effectively defines the output dimensions. Applying pooling with a stride s is equivalent to applying the pooling with no stride (i.e., $s = 1$) and then sub-sampling by a factor of s. It is common to add zero padding to $\mathbf{z}$ such that its length is divisible by s.

Another very common nonlinear function is the *softmax*, which normalizes vectors into probabilities. The output of the model, the embedding, may undergo an additional linear layer to transform it to a vector of size of $1 \times N$, termed *logits*, where N is the number of classes. The logits, here denoted as $\mathbf{v}$, are the input to the softmax operation defined as follows:

$$\text{softmax}(v_i) = \frac{e^{v_i}}{\sum_{j=1}^{N} e^{v_j}}, \qquad i \in [1, ..., N]. \tag{2}$$

3 Loss Functions

Defining the loss function of the model, denoted as $\mathcal{L}$, is critical and usually chosen based on the characteristics of the dataset and the task at hand. Though datasets can vary, tasks performed by NN models can be divided into two coarse groups: (1) regression tasks and (2) classification tasks.

A *regression* problem aims at approximating a mapping function from input variables to a continuous output variable(s). For NN tasks, the output of the network should predict a continuous value of interest. Common NN regression problems include image denoising [212], deblurring [124], inpainting [203], and more. In these tasks, it is common to use the Mean Squared Error (MSE), Structural SIMilarity (SSIM), or ℓ_1 loss as the loss function. The MSE (ℓ_2 error) imposes a larger penalty for larger errors, compared to the ℓ_1 error which is more robust to outliers in the data. The SSIM and its multiscale version [214] help improving the perceptual quality.

In the *classification* task, the goal is to identify the correct class of a given sample from predefined N classes. A common loss function for such tasks is the *cross-entropy* loss. It is implemented based on a normalized vector of probabilities corresponding to a list of potential outcomes. This normalized vector is calculated by the softmax nonlinear function (Eq. 2). The cross-entropy loss is defined as

$$\mathcal{L}_{CE} = -\sum_{i=1}^{N} y_i \log(p_i), \tag{3}$$

where y_i is the ground-truth probability (the label) of the input to belong to class i and p_i is the model prediction score for this class. The label is usually binary, i.e., it contains 1 in a single index (corresponding to the true class). This type of representation is known as *one-hot encoding*. The class is predicted in the network by selecting the largest probability and the log-loss is used to increase this probability.

Notice that a network may provide multiple outputs per input data point. For example, in the problem of image semantic segmentation, the network predicts a class for each pixel in the image. In the task of object detection, the network

outputs a list of objects, where each one is defined by a bounding box (found using a regression loss) and a class (found using a classification loss). Section 7.1 details these different tasks. Since in some problems, the labelled data are imbalanced, one may use weighted softmax (that weigh less frequent classes) or the focal loss [108].

3.1 Metric Learning

An interesting property of the log-loss function used for classification is that it implicitly clusters classes in the network embedding space during training. However, for a clustering task, these vanilla distance criteria often produce unsatisfactory performance as different class clusters can be positioned closely in the embedding space and may cause miss-classification for samples that do not reside in the specific training set distribution.

Therefore, different metric learning techniques have been developed to produce an embedding space that brings closer intra-class samples and increases inter-class distances. This results in better accuracy and robustness of the network. It allows the network to be able to distinguish between two data samples if they are from the same class or not, just by comparing their embeddings, even if their classes have not been present at training time.

Metric learning is very useful for tasks such as face recognition and identification, where the number of subjects to be tested is not known at training time and new identities that were not present during training should also be identified/recognized (e.g., given two images, the network should decide whether these correspond to the same or different persons).

An example for a popular metric loss is the *triplet loss* [153]. It enforces a margin between instances of the same class and other classes in the embedding feature space. This approach increases performance accuracy and robustness due to the large separation between class clusters in the embedding space. The triplet loss can be used in various tasks, namely detection, classification, recognition, and other tasks of an unknown number of classes.

In this approach, three instances are used in each training step i: an anchor $\mathbf{x}_i^a$, another instance $\mathbf{x}_i^p$ from the same class of the anchor (positive sample), and a sample $\mathbf{x}_i^n$ from a different class (negative class). They are required to obey the following inequality:

$$\left\| \Phi(\mathbf{x}_i^a) - \Phi(\mathbf{x}_i^p) \right\|_2^2 + \alpha < \left\| \Phi(\mathbf{x}_i^a) - \Phi(\mathbf{x}_i^n) \right\|_2^2, \tag{4}$$

where $\alpha > 0$ enforces the wanted margin from other classes. Thus, the triplet loss is defined by

$$\mathcal{L} = \sum_i \left\| \Phi(\mathbf{x}_i^a) - \Phi(\mathbf{x}_i^p) \right\|_2^2 - \left\| \Phi(\mathbf{x}_i^a) - \Phi(\mathbf{x}_i^n) \right\|_2^2 + \alpha. \tag{5}$$

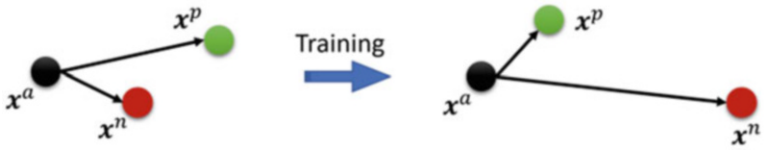


Fig. 7 Triplet loss: minimizes the distance between two similar class examples (anchor and positive) and maximizes the distance between two different class examples (anchor and negative)

Figure 7 presents a schematic illustration of the triplet loss influence on samples in the embedding space. This illustration also exhibits a specific triplet example, where the positive examples are relatively far from the anchor while negative examples are relatively near the anchor. Finding such examples that violate the triplet condition is desirable during training. They may be found by online or offline searches known as *hard negative mining*. A preprocessing of the instances in the embedding space is performed to find violating examples for training the network.

Finding the "best" instances for training can, evidently, aid in achieving improved convergence. However, searching for them is often time consuming, and therefore alternative techniques are being explored.

An intriguing metric learning approach relies on "classification"-type loss functions, where the network is trained given a fixed number of classes. Yet, these losses are designed to create good embedding space that creates margin between classes, which in turn provides good prediction of similarity between two inputs. Popular examples include the Cos-loss [192], Arc-loss [33], and SphereFace [113].

4 Neural Network Training

Given a loss function, the weights of the neural network are updated to minimize it for a given training set. The training process of a neural network requires a large database due to the nature of the network (structure and amount of parameters) and GPUs for efficient training implementation.

In general, training methods can be divided into supervised and unsupervised training. The former, which has received a greater attention in the research community, consists of labeled data that are usually very expensive and time consuming to obtain, whereas the latter does not assume known ground-truth labels and applies to the more common case in real-world scenarios where the majority of the data available for training are not annotated. Note though that supervised training usually achieves significantly better network performance compared to the unsupervised case. Therefore, a lot of resources are invested in labeling datasets for training. Thus, we focus in this chapter mainly on the supervised setting.

In neural networks, regardless of the model task, all training phases have the same goal: to minimize a predefined error function, also denoted as the loss/cost function.

This is done in two stages: (a) a feed-forward pass of the input data through all the network layers, calculating the error using the predicted outputs and their ground-truth labels (if available), followed by (b) backpropagation of the errors through the network to update their weights, from the last layer to the first. This process is performed continuously to find the optimized values for the weights of the network.

The backpropagation algorithm provides the gradients of the error with respect to the network weights. These gradients are used to update the weights of the network. Calculating them based on the whole input data is computationally demanding, and therefore, the common practice is to use subsets of the training set, termed *mini-batches*, and cycle over the entire training set multiple times. Each cycle of training over the whole dataset is termed an *epoch*, and in every cycle the data samples are used in a random order to avoid biases. The training process ends when convergence in the loss function is obtained. Since most NN problems are not convex, an optimal solution is not assured. We turn now to describe in more detail the training process using backpropagation.

4.1 Backpropagation

The backpropagation process is performed to update all the parameters of the model, with the goal of decreasing the loss function value. The process starts with a feed-forward pass of input data, $\mathbf{x}$, through all the network layers. After which the loss function value is calculated and denoted as $\mathcal{L}(\mathbf{x}, \mathbf{W})$, where $\mathbf{W}$ are the model parameters (including the model weights and biases, for formulation convenience). Then, the backpropagation is initiated by computing the value of $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$, followed by the update of the network weights. All the weights are updated recursively by calculating the gradients of every layer, from the final one to the input layer, using the chain rule.

Denote the output of layer l as $\mathbf{z}^{(l)}$. Following the chain rule, the gradients of a given layer l with parameters $\mathbf{W}^{(l)}$ with respect to its input $\mathbf{z}^{(l)}$ are

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(l-1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l)}(\mathbf{W}^{(l)}, \mathbf{z}^{(l-1)})}{\partial \mathbf{z}^{(l-1)}}, \tag{6}$$

and the gradients with respect to the parameters are

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l)}(\mathbf{W}^{(l)}, \mathbf{z}^{(l-1)})}{\partial \mathbf{W}^{(l)}}. \tag{7}$$

These two formulas of the backpropagation algorithm dictate the gradients calculation with respect to the parameters for each layer in the network, and, therefore, the optimization can be performed using gradient-based optimizers (see Sect. 5 for more details).

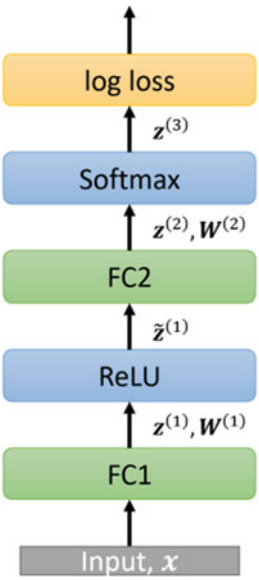


Fig. 8 Simple classification model example, consisting of a two-layered fully connected model

To demonstrate the use of the backpropagation technique for the calculation of the network gradients, we turn to consider an example of a simple classification model with two layers: a fully connected layer with a ReLU activation function followed by another fully connected layer with softmax function and log-loss. See Fig. 8 for the model illustration.

Denote by $\mathbf{z}^{(3)}$ the output of the softmax layer, and assume that the input $\mathbf{x}$ belongs to class k (using one-hot encoding $y_k = 1$). The log-loss in this case is

$$\mathcal{L} = -\sum_i \log\left(z_i^{(3)}\right) y_i = -\log\left(\frac{\exp\left(z_k^{(2)}\right)}{\sum_i \exp\left(z_i^{(2)}\right)}\right) = -z_k^{(2)} + \log\left(\sum_j \exp z_j^{(2)}\right). \tag{8}$$

For all $i \neq k$, the gradient of the error with respect to the softmax input $z_i^{(2)}$ is

$$\frac{\partial \mathcal{L}}{\partial z_i^{(2)}} = \frac{\exp\left(z_i^{(2)}\right)}{\sum_j \exp\left(z_j^{(2)}\right)} \equiv g_i\,. \tag{9}$$

Notice that this implies that we need to decrease the value of $z_i^{(2)}$ (the ith logit) proportionally to the probability the network provides to it, while for the correct label, $i = k$, the derivative is

$$\frac{\partial \mathcal{L}}{\partial z_k^{(2)}} = -1 + \frac{\exp\left(z_k^{(2)}\right)}{\sum_j \exp\left(z_j^{(2)}\right)} = g_k - 1, \tag{10}$$

which implies that the value of the logit element associated with the true label should be increased proportionally to the mistake the network is currently doing in the prediction.

The output $\mathbf{z}^{(2)}$ is a product of a fully connected layer. Therefore, it can be formulated as follows:

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\tilde{\mathbf{z}}^{(1)}, \tag{11}$$

where $\tilde{\mathbf{z}}^{(1)}$ is the output of the ReLu function. Following the backpropagation rules, we get that for this layer, the derivative with respect to its input is

$$\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{z}}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(2)}} \cdot \frac{\partial \mathbf{z}^{(2)}(\mathbf{W}^{(2)}, \tilde{\mathbf{z}}^{(1)})}{\partial \tilde{\mathbf{z}}^{(1)}} = \mathbf{W}^{(2)} \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(2)}}, \tag{12}$$

whereas the derivative with respect to its parameters is

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(2)}} \cdot \frac{\partial \mathbf{z}^{(2)}(\mathbf{W}^{(2)}, \tilde{\mathbf{z}}^{(1)})}{\partial \mathbf{W}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(2)}} \cdot \left(\tilde{\mathbf{z}}^{(1)}\right)^T. \tag{13}$$

The ReLU operation has no weight to update but affects the gradients. The derivative of this stage follows

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{z}}^{(1)}} \cdot \frac{\partial \tilde{\mathbf{z}}^{(1)}(\mathbf{W}^{(1)}, I)}{\partial \mathbf{z}^{(1)}} = \begin{cases} 0, & \text{if } \mathbf{z}^{(1)} < 0 \\ \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{z}}^{(1)}}, & \text{otherwise.} \end{cases} \tag{14}$$

The final derivative with respect to the input $\partial \mathcal{L}/\partial \mathbf{x}$ is calculated similar to Eq. (12).

4.2 Training Considerations

There are several considerations that should be addressed when training an NN. The most infamous is the *overfitting*, i.e., when the model too closely fits to the training dataset but does not generalize well to the test set. When this occurs, high training data precision is achieved, while the precision on the test data (not used during training) is low [183]. For this purpose, various regularization techniques have been proposed. We discuss some of them in Sect. 6.

A second consideration is the vanishing/exploding gradients occurring during training. Vanishing gradients are a result of multiplications with values smaller than one during their calculation in the backpropagation recursion. This can be resolved using activation functions and batch normalization detailed in Sect. 6. On the other hand, the gradients might also explode due to derivatives that are significantly larger than one in the backpropagation calculation. This makes the training unstable and

may imply the need for redesigning the model (e.g., replace a vanilla RNN with a gated architecture such as LSTMs) or the use of gradient clipping [132].

Another important issue is the requirement that the training dataset must represent the true distribution of the task at hand. This usually enforces very large, annotated datasets, which necessitate significant funding and manpower to obtain. In this case, considerable efforts must be invested to train the network using these large datasets, commonly with multiple GPUs for several days [93, 85]. One may use techniques such as domain adaptation [198] or transfer learning [180] to use already existing networks or large datasets for new tasks.

4.3 Hyper-Parameter Tuning

While ongoing research studies the effect of hyper-parameter values during neural network training, there are several "rules of thumb" that can aid practitioners in achieving optimal model convergence.

The first relates to the *learning rate* initialization and training schedule. Commonly, an adaptive learning rate is used in which its values are decreased when no improvement in the loss is evident for a pre-determined number of epochs. In addition to this strategy, there are some more evolved methods for the learning rate policy. For example, cycle learning rate (CLR) [167] and one-cycle learning rate (OCL) [168] have shown to help models to converge faster and improve their performance. In the CLR case, a learning rate range is defined, and during training the learning rate value varies cyclically within this range. While increasing the learning rate might harm the convergence in the short term, it is shown to be beneficial to the final model state. The OCL suggests performing only **one** cycle of increasing–decreasing of the learning rate through the entire training phase, while the last epochs are performed with an extremely low learning rate. This policy has shown to be effective since the learning rate is the highest in the middle of the training and acts as a regularization to prevent overfitting.

The parameters of *weight decay* and *dropouts* are also important to tune in terms of improving convergence and accuracy, as detailed in Sect. 6. Note that their hyper-parameter values can be dependent also on the policy determined for the learning rate [167].

Regarding the architecture of a network, an interesting research question is about the effectiveness of the model depth or the trade-off between network depth and width (each layer's size). This has been studied from various perspectives, e.g., from an Information Bottleneck point of view [163], or via studying the generalization property of the model [23], where in both cases it has been shown that deeper models exhibit better generalization and faster convergence. A recent work [182] empirically studied the optimal ratio between the sizes of the input data, the width of the network, and its depth.

5 Training Optimizers

Training neural networks is done by applying an optimizer to reach an optimal solution for the defined loss function. Its goal is to find the parameters of the model, e.g., weights and biases, which achieve minimum error for the training set samples: $(\mathbf{x}_i, y_i)$, where y_i is the label for the instance $\mathbf{x}_i$. For a loss function $\mathcal{L}(\cdot)$, the objective reads as

$$\sum_i \mathcal{L}(\Phi(\mathbf{x}_i, \mathbf{W}), y_i), \tag{15}$$

for ease of notation, all model parameters are denoted as $\mathbf{W}$. A variety of optimizers have been proposed and implemented for minimizing Eq. (15). Yet, due to the size of the network and training dataset, mainly first-order methods are being considered, i.e., strategies that rely only on the gradients (and not on second-order derivatives such as the Hessian).

Several gradient-based optimizers are commonly used for updating the parameters of the model. These NN parameters are updated in the opposite direction of the objective function's gradient, $g_{\{\mathrm{GD},\mathcal{T}(t)\}}$, where $\mathcal{T}(t)$ is a randomly chosen subgroup of size $n' < n$ training samples used in iteration t (n is the size of the training dataset). Namely, at iteration t, the weights are calculated as

$$\mathbf{W}(t) = \mathbf{W}(t-1) - \eta \cdot g_{\{\mathrm{GD},\mathcal{T}(t)\}}, \tag{16}$$

where η is the learning rate that determines the size of the steps taken to reach the (local) minimum and the gradient step, and $g_{\{\mathrm{GD},\mathcal{T}(t)\}}$ is computed using the samples in $\mathcal{T}(t)$ as

$$g_{\{\mathrm{GD},\mathcal{T}(t)\}} = \frac{1}{n'} \sum_{i \in \mathcal{T}(t)} \nabla_W \mathcal{L}(\mathbf{W}(t); \mathbf{x}_i; y_i), \tag{17}$$

where the pair $(\mathbf{x}_i, y_i)$ is a training example and its corresponding label in the training set, and $\mathcal{L}$ is the loss function. However, needless to say that calculating the gradient on the whole dataset is computationally demanding. To this end, Stochastic Gradient Descent (SGD) is more popular, since it calculates the gradient in Eq. (17) for only one randomly chosen example from the data, i.e., $n' = 1$.

Since the update by SGD depends on a different sample at each iteration, it has a high variance that causes the loss value to fluctuate. While this behavior may enable it to jump to a new and potentially better local minima, it might ultimately complicate convergence, as SGD may keep overshooting. To improve convergence and exploit parallel computing power, mini-batch SGD is proposed in which the gradient in Eq. (17) is calculated with $n' > 1$ (but not all the data).

An acceleration in convergence may be obtained by using the history of the last gradient steps, in order to stabilize the optimization. One such approach uses

adaptive momentum instead of a fixed step size. This is calculated based on exponential smoothing on the gradients, i.e.,

$$\begin{aligned} M(t) &= \beta \cdot M(t-1) + (1-\beta) \cdot g_{\{\mathrm{SGD},\mathcal{T}(t)\}}, \\ \mathbf{W}(t) &= \mathbf{W}(t-1) - \eta M(t), \end{aligned} \tag{18}$$

where $M(t)$ approximates the first moment of $g_{\{\mathrm{SGD},\mathcal{T}(t)\}}$. A typical value for the constant is $\beta \sim 0.9$, which implies taking into account the last 10 gradient steps in the momentum variable $M(t)$ [137]. A well-known variant of momentum proposed by Nestrov et al. [125] is the Nestrov Accelerated Gradient (NAG). It is similar to momentum but calculates the gradient step as if the network weights have been already updated with the current momentum direction.

Another popular technique is the Adaptive Moment Estimation (ADAM) [89], which also computes adaptive learning rates. In addition to storing an exponentially decaying average of past squared gradients, $V(t)$, ADAM also keeps an exponentially decaying average of past gradients, $M(t)$, in the following way:

$$\begin{aligned} M(t) &= \beta_1 M(t-1) + (1-\beta_1) g_t, \\ V(t) &= \beta_2 V(t-1) + (1-\beta_2) g_t^2, \end{aligned} \tag{19}$$

where g_t is the gradient of the current batch, β_1 and β_2 are ADAM's hyperparameters, usually set to 0.9 and 0.999, respectively, and $M(t)$ and $V(t)$ are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients, respectively, hence the name of the method—Adaptive Moment Estimation. As $M(t)$ and $V(t)$ are initialized as vectors of 0s, the authors of ADAM observe that they are biased toward zero, especially during the initial timesteps. To counteract these biases, bias-corrected first and second moments are used: $\hat{M}(t) = M(t)/(1-\beta_1(t))$ and $\hat{V}(t) = V(t)/(1-\beta_2(t))$. Therefore, the ADAM update rule is as follows:

$$\mathbf{W}(t+1) = \mathbf{W}(t) - \frac{\eta}{\sqrt{\hat{V}(t)+\epsilon}} \hat{M}(t). \tag{20}$$

ADAM has two popular extensions: AdamW by Loshchilov et al. [116] and AMSGrad by Redddi et al. [141]. There are several additional common optimizers that have adaptive momentum, such as AdaGrad [41], AdaDelta [210], or RMSprop [29]. It must be noted that since the NN optimization is non-convex, the minimal error point reached by each optimizer is rarely the same. Thus, speedy convergence is not always favored. In particular, it has been observed that momentum leads to better generalization than ADAM, which usually converges faster [88]. Thus, the common practice is to make the development with ADAM and then make the final training with momentum.

6 Training Regularizations

One of the great advantageous of NN is their ability to generalize, i.e., correctly predict unseen data [78]. This must be ensured during the training process and is accomplished by several regularization methods, detailed here. The most common are weight decay [94], dropout [172], batch normalization [75], and the use of data augmentation [160].

Weight decay is a basic tool to limit the growth of the weights by adding a regularization term to the cost function for large weights, which is the sum of squares of all the weights, i.e., $\sum_i |W_i|^2$.

The key idea in *dropout* is to randomly drop units (along with their connections) from the neural network during training and thus prevent units from co-adapting too much. The percentage of dropped units is critical since a large amount will result in poor learning. The common values are 20%–50% dropped units.

Batch normalization is a mean to deal with changes in the distribution of the model's parameters during training. The layers need to adapt to these (often noisy) changes between instances during training. Batch normalization causes the features of each training batch to have a mean of 0 and a variance of 1 in the layer it is being applied. To normalize a value across a batch, i.e., to batch normalize the value, the batch mean, μ_B, is subtracted and the result is divided by the batch standard deviation, $\sqrt{\sigma_B^2 + \epsilon}$. Note that a small constant ϵ is added to the variance in order to avoid dividing by zero. The batch normalizing transform of a given input, $\mathbf{x}$, is

$$\mathrm{BN}(\mathbf{x}) = \gamma\left(\frac{\mathbf{x} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\right) + \beta. \tag{21}$$

Notice the (learnable) scale and bias parameters γ and β, which provides the NN with freedom to deviate from the zero mean and unit variance. BN is less effective when used with small batch sizes since in this case the statistics calculated per each is less accurate. Thus, techniques such as group normalization [200] or Filter Response Normalization (FRN) [166] have been proposed.

Data augmentation is a very common strategy used during training to artificially "increase" the size of the training data and make the network robust to transformations that do not change the input label. For example, in the task of classification, a shifted cat is still a cat; see Fig. 9 for more similar augmentation. In the task of denoising, flipped noisy input should result in a flipped clean output. Thus, during training, the network is also trained with the transformed data to improve its performance.

Common augmentations are randomly flipping, rotating, scaling, cropping, translating, or adding noise to the data. Other more sophisticated techniques that lead to a significant improvement in network performance include mixup [211], cutout [37], and augmentations that are learned automatically [26, 107, 27].

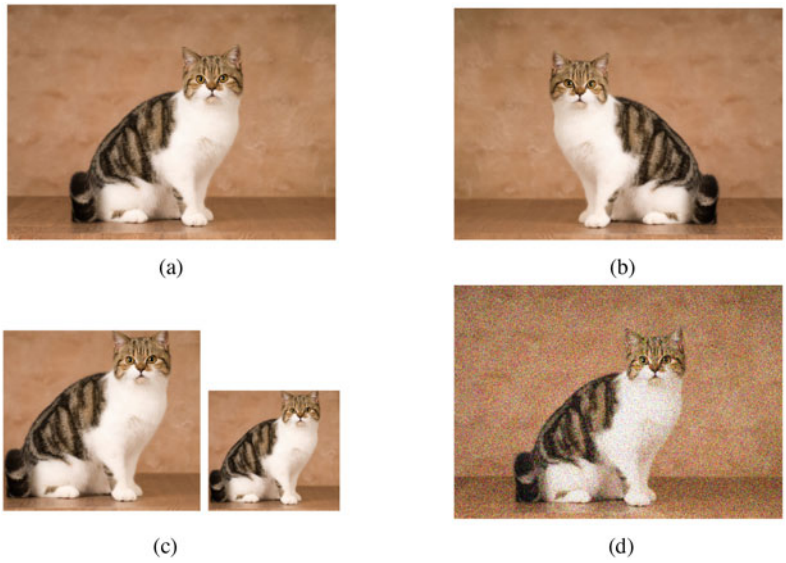


Fig. 9 Different image augmentations. (**a**) Original image. (**b**) Flip augmentation. (**c**) Crop and scale augmentation. (**d**) Noise augmentation

7 Advanced NN Architectures

The basic building blocks, which compose the NN model architecture, are used in frequently innovative structures. In this section, such known architectures with state-of-the-art performance are presented, divided by tasks and data types: detection and segmentation tasks are described in Sect. 7.1, sequential data handling is elaborated in Sect. 7.2, and processing data on irregular grids is presented in Sect. 7.3. Clearly, there are many other use-cases and architectures, which are not mentioned here.

7.1 Deep Learning for Detection and Segmentation

Many research works focus on detecting multiple objects in a scene, due to its numerous applications. This problem can be divided into four sub-tasks as follows, where we refer here to image datasets although the same concept can be applied to different domains as well:

1. *Classification and localization*: The main object in the image is detected and then localized by a surrounding bounding box and classified from a pre-known set.
2. *Object detection*: Detection of all objects in a scene that belong to a pre-known set and then classifying and providing a bounding box for each of them.

3. *Semantic segmentation*: Partitioning the image into coherent parts by assigning each pixel in the image with its own classification label (associated with the object the pixel belongs to), for example, having a pixel-wise differentiation between animals, sky, and background (generic class for all object that no class is assigned to) in an image.
4. *Instance segmentation*: Multiple objects segmentation and classification from a pre-known set (similar to object detection but for each object all its pixels are identified instead of providing a bounding box for it).

Today, state-of-the-art object detection performance is achieved with architectures such as Faster-RCNN [147, 195], You Only Look Once (YOLO) [142, 143, 144], Single Shot Detector (SSD) [112], and Fully Convolutional One-Stage Object Detection (FCOS) [184]. The object detection models provide a list of detected bounding boxes with the class of each of them.

Segmentation tasks are mostly implemented using fully convolutional network. Known segmentation models include UNet [148], Mask-RCNN [62], and Deeplab [14]. These architectures have the same input/output spatial size since the output represents the segmentation map of the input image.

Both object detection and segmentation tasks are analyzed via the Intersection over Union (IoU) metric. The IoU is defined as the ratio between the intersection area of the object's ground-truth pixels, B_g, with the corresponding predicted pixels, B_p, and the union of these groups of pixels. The IoU is formulated as

$$\text{IoU} = \frac{\text{Area}\{B_g \cap B_p\}}{\text{Area}\{B_g \cup B_p\}}. \tag{22}$$

As this measure evaluates only the quality of the bounding box, a mean Average Precision (mAP) is commonly used to evaluate the models' performance. The mAP is defined as the ratio of the correctly detected (or segmented) objects, where an object is considered to be detected correctly if there is a bounding box for it with the correct class and an IoU greater than 0.5 (or another specified constant).

Another common evaluation metric is the F1 score, which is the harmonic average of the precision and the recall values. See Eq. (24) below. They are calculated using the following definitions that are presented for the case of semantic segmentation:

- True Positive (TP): the predicted class of a pixel matches it ground-truth label
- False Positive (FP): the predicted pixel of an object was falsely determined
- False Negative (FN): a ground-truth pixel of an object was not predicted

Now that they are defined, the *precision*, *recall*, and F1 are given by

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{23}$$

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{24}$$

7.2 Deep Learning on Sequential Data

Sequential data are composed of time-sensitive signals such as the output of different sensors, audio recordings, NLP sentences, or any signal that its order is of importance. Therefore, these data must be processed accordingly.

Initially, sequential data were processed with Recurrent NN (RNN) [77] that has recurrent (feedback) connections, where outputs of the network at a given time-step serve as input to the model (in addition to the input data) at the next time-step. This introduces the time-dependent feature of the NN. An RNN is illustrated in Fig. 4.

However, it was quickly realized that during training, vanilla RNNs suffer from vanishing/exploding gradients. This phenomenon, originated from the use of finite-precision backpropagation process, limits the size of the sequence.

To this end, a corner stone block is used: the Long-Short-Term Memory (LSTM [68]). Mostly used for NLP tasks, the LSTM is an RNN block with gates. During training, these gates learn which part of the sentence to forget or to memorize. The gating allows some of the gradients to backpropagate unchanged, which aids the vanishing gradient symptom. Notice that RNNs (and LSTMs) can process a sentence in a bidirectional mode, i.e., process a sentence in two directions, from the beginning to the end and vice versa. This mechanism allows a better grasp of the input context by the network. Examples for popular research tasks in NLP data include question answering [139], translation [98], and text generation [56].

Sentences Processing An important issue in NLP is representing words in preparation to serve as network input. The use of straightforward indices is not effective since there are thousands of words in a language. Therefore, it is common to process text data via *word embedding*, which is a vector representation of each word in some fixed dimension. This method enables to encapsulate relationships between words.

Word2Vec is a classic methodology to calculate word embeddings [119]. In this approach, these vector representations are calculated using an NN model that learns their context. More advanced options for creating efficient word representations include BERT [36], ELMO [133], RoBERTa [114], XLNet [206], and GPT [140].

Audio Processing Audio recordings are used for multiple interesting tasks, such as speech to text, text to speech, and speech processing. In the audio case, the common input to speech systems is the Mel Frequency Cepstral Coefficient (MFCC) or a Short-Time Fourier Transform (STFT) image, as opposed to the audio raw data. A milestone example for speech processing NN architecture is the *WaveNet* [129]. This architecture is an autoregressive model that synthesizes speech or audio signals. It is based on dilated convolutional layers that have large receptive fields that allow efficient processing. Another prominent synthesis model for sequential data is the Tacotron [157].

The Attention Model As mentioned in Sect. 2, one may use RNN for translation using the encoder decoder model, which encodes a source sentence into a vector, which is then decoded to a target language. Instead of relying on a compressed vector, which may lose information, the *attention models* learn where or what to

focus on from the whole input sequence. Introduced in 2015 [3], attention models have shown superior performance over encoder–decoder architectures in tasks such as translation, text to speech, and image captioning. Recently, it has been suggested to replace the recurrent network structure totally by the attention mechanism, which results with the *transformers network* models [187].

7.3 Deep Learning on Irregular Grids

A wide variety of data acquisition mechanisms do not represent the data on a grid as is common with images data. A prominent example is 3D imaging (e.g., using LIDAR), where the input data are represented as points in a 3D space with or without color information. Processing such data is not trivial as standard network components, such as convolutions, assume a grid of the data. Therefore, they cannot be applied as is and custom operations are required. We focus our discussion here on the case of NN for 3D data.

Today, real-time processing of 3D scenes can be achieved with advanced NN models that are customized to these irregular grids. The different processing techniques for these irregular grid data can be divided by the type of representation used for the data:

1. **Points processing.** 3D data points are processed as points in space, i.e., a list of the point coordinates is given as the input to the NN. A popular network for this representation is *PointNet* [135]. It is the first to efficiently achieve satisfactory results directly on the point cloud. Yet, it is limited by the number of points that can be analyzed, computational time and performance. Some more recent models that improve its performance include PointNet++ [136], PointCNN [103], and DGCNN [196]. Strategies to improve its efficiency have been proposed in learning to sample [39] and RandLA-Net [71]. A hierarchical Gaussian mixture-based point cloud network has been proposed in PointGMM [65]. Another recent useful representation for point clouds is the signed distance function [131].
2. **Multi-view 2D projections.** 3D data points are projected (from various angles) to the 2D domain so that known 2D processing techniques can be used [104, 83].
3. **Volumetric (voxels).** 3D data points are represented in a grid-based *voxel* representation. This is analogous to a 2D representation and is therefore advantageous. However, it is computationally exhaustive [201] and losses resolution.
4. **Meshes.** Mesh represents the 3D domain via a graph that defines the connectivity between the different points. Yet, this graph has a special structure such that it creates the surface of the 3D shape (in the common case of triangular mesh, the shape surface is presented by a set of triangles connected to each other). In 2015, Masci et al. [7] have shown it is possible to learn features using

DL on meshes. Since then, a significant advancement has been made in mesh processing [58, 123, 111, 64, 19].

5. **Graphs.** Graph representations are common for representing nonlinear structured data. Some works have proposed efficient NN models for 3D data points on a grid-based graph structure [30, 91, 126, 173].

8 Summary

This chapter provided a general survey of the basic concepts in neural networks. As this field is expanding very fast, the space is too short to describe all the developments in it, even though most of them are from the past eight years. Yet, we briefly mention here few important problems that are currently being studied.

1. **Domain adaptation and transfer learning.** As many applications necessitate data that are very difficult to obtain, some methods aim at training models based on scarce datasets. A popular methodology for dealing with insufficient annotated data is *domain adaptation*, in which a robust and high performance NN model, trained on a source distribution, is used to aid the training of a similar model (usually with the same goal, e.g., in classification the same classes are searched for) on data from a target distribution that are either unlabeled or small in number [47, 130, 162]. An example is adapting an NN trained on simulation data to real-life data with the same labels [186, 69]. On a similar note, *transfer learning* [180, 38] can also be used in similar cases, where in addition to the difference in the data the input and output tasks are not the same but only similar (in domain adaptation the task is the same and only the distributions are different). One such example is using a network trained on natural images to classify medical data [4].
2. **Few-shot learning.** A special case of learning with small datasets is *few-shot learning* [197], where one is provided either with just semantic information of the target classes (zero-shot learning), only one labelled example per class (one-shot learning), or with just few samples (general few-shot learning). Approaches developed for these problems have shown great success in many applications, such as image classification [176, 155, 174], object detection [84], and segmentation [10].
3. **Online learning.** Various deep learning challenges occur due to new distributions or class types introduced to the model during a continuous operation of the system (post-training) and now must be learnt by the model. The model can update its weights to incorporate these new data using *online learning* techniques. There is a need for special training in this case, as systems that just learn based on the new examples may suffer from a reduced performance on the original data. This phenomenon is known as catastrophic forgetting [87]. Often, the model tends to forget the representation of part of the distribution it already learned, and thus it develops a bias toward the new data. A specific

example of online learning is *incremental learning* [11], where the new data is of different classes than the original ones.

4. **AutoML.** When approaching real-life problems, there is an inherent pipeline of tasks to be performed before using DL tools, such as problem definition, preparing the data, and processing it. Commonly, these tasks are performed by specialists and require deep system understating. To this end, the *autoML* paradigm attempts to generalize this process by automatically learning and tuning the model used [44].
   A particular popular task in autoML is *Neural Architecture Search (NAS)* [42]. This is of interest since the NN architecture restricts its performance. However, searching for the optimal architecture for a specific task, and from a set of predefined operations, is computationally exhaustive when performed in a straight forward manner. Therefore, ongoing research attempts to overcome this limitation. An example is the DARTS [110] strategy and its extensions [127, 18] where the key contribution is finding, in a differentiable manner, the connections between network operations that form an NN architecture. This framework decreases the search time and improves the final accuracy.
5. **Reinforcement Learning.** To date, the most effective training method for decision-based actions, such as robot movement and video games, is *Reinforcement Learning* (RL) [81, 122, 178]. In RL, the model tries to maximize some predefined award score by learning which action to take, from a set of defined actions in specific scenarios.
6. **Diffusion Models.** Diffusion probabilistic models are a non-adversarial alternative to GANs, inspired by nonequilibrium thermodynamics. These models generate an image output from noise [67, 31] in the following manner similar to the autoencoders regime: in the forward diffusion pass, noise is added T times, while in the reverse diffusion pass, a neural network denoises these images sequentially. Diffusion models are commonly used in tasks such as image denoising, inpainting, super-resolution, and image generation. One of the leading examples of a diffusion model application is image generation, where it has been shown to produce state-of-the-art results in recent years [67]. Prominent examples where diffusion models are used for generation are DALL-E-2, Midjourney and Stable-Diffusion.
7. **Large Language Models (LLMs)** LLMs such as GPT-3 [9, 46] are deep learning models that can process and generate natural language text at an unprecedented scale. These models have been shown to perform tasks such as language translation, text summarization, and even creative writing with impressive results [9]. One example of an LLM is ChatGPT, a large language model that can converse with users and provide relevant responses to their inputs. The potential impact of LLMs is significant, as they have the ability to assist humans in tasks such as writing, research, and even customer service, potentially leading to increased efficiency and productivity in various industries. However, concerns have also been raised about the ethical and societal implications of these models, including issues related to bias, privacy, and ownership [6].

Table 2 List of (selected) commonly used DL techniques and their description

Task	Technique	Description
Classification	VGG	Very deep CNN used for large-scale image classification [165]
	ResNet	A residual learning framework for training a substantially deep model. Each layer consists of a residual function with reference to the layer inputs [63]
	DenseNet	A dense CNN that connects each layer to every other layer in a feed-forward fashion [72]
	EfficientNet	A scalable CNN architecture that can vary its size depending on the current setup limitations [182]
Image Restoration	DenoiseNet	A deep network for image and video denoising [146]
	DeepISP	A deep network for end-to-end image restoration. It performs jointly denoising, demosaicing, and color enhancement [154]
	CycleISP	A deep technique for denoising of realistic noise [209]
	DBPN/RCAN	Deep networks for image super-resolution trained for a bicubic downsampling kernel [59, 213]. It is possible to extend their use to any kernel by applying a correction filter [74]
	DeblurGAN	GAN-based deep blind motion deblurring [96]
	EDVR	Deep network for video restoration [194].
Language modeling (see Sect. 7.2)	BERT	The Bidirectional Encoder Representations from Transformers (BERT) is used for multiple purposes in language modeling. BERT learns a language modeling which is then used as the basis for different NLP models [35]
	GPT	It is a causal (unidirectional) transformer for language modeling pre-trained on a large corpus [9, 46, 140]
	XLnet	A generalized autoregressive pretraining method for language modeling that enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order [207]
	Elmo	Embeddings from Language Models (ELMo) is a deep bidirectional language model that learns both complex characteristics of word usage and how these usages vary across linguistic contexts [134]
Encoder–Decoder (see Sect. 2)	Autoencoder	AE architecture is a (usually symmetric) model, composed of an encoder–decoder, used to learn efficient data coding. It is used for several tasks such as image denoising [146], image captioning [191], feature extraction [190], and segmentation [2]
	Variational Autoencoder	An AE architecture that encodes the input as a distribution over the latent space [90], Usually leads to better latent representations than regular AE
	U-Net	A popular U-shaped encoder–decoder network that is very useful for segmentation and image restoration tasks [148]
	Neural inverse rendering	AE architecture designed to generate an image using a graphical engine acting as the decoder [95]

GAN	SN-GAN/BigGAN	conditional Generative Adversarial Networks (GANs) used on largest scale images (specifically ImageNet's 128×128 pixels images) are trained with orthogonal regularization to the generator and using Spectral Normalization [121, 8]
	SinGAN	An unconditional generative model that can be learned from a single natural image. Its architecture consists of a pyramid of fully convolutional GANs that allow generating new patches of arbitrary size and aspect ratio [156]
	StyleGAN	StyleGAN introduces a new generator architecture inspired from the style transfer task. It has shown improvement in many quality metrics compared to a traditional GAN [86]
Face recognition	FaceNet	A CNN architecture that directly learns a mapping from face images to an embedding space using triplet loss [153]
	Deep-face	A nine-layer neural network explicitly exploiting a 3D face modeling in order to derive a face representation. The network consists of more than 120 million parameters using several locally connected layers without weight sharing, rather than the standard convolutional layers [179]
	CosFace	This paper tackles the face recognition task by defining a Large Margin Cosine Loss (LMCL) by reformulating the softmax loss as a cosine loss by L2 normalizing both features and weight vectors to remove radial variations. This enables the definition of a cosine margin term to further maximize the decision margin in the angular space [193]
	ArcFace	ArcFace has shown impressive performance by training using an Additive Angular Margin Loss which corresponds to a geodesic distance on a hypersphere [34]
Speech processing	Wavenet	This architecture is an autoregressive model that synthesizes speech or audio signals. It is based on dilated convolutional layers that have large receptive fields that allow efficient processing [129]
	Tacotron	A synthesis model for sequential data is the [157]
	ESPnet-TTS	Extensive toolkit for end-to-end text-to-speech processing. It also provides pre-trained models and samples of all of the recipes so that users can use it as a baseline [60]
Image to image translation	pix2pix	A conditional GAN used for multiple tasks such as at synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images [76]
	Cycle GAN	A model to learn from unpaired data. It comprises two GAN models with interchanging inputs and a loss that enforces cycle consistency (translating an image from type A to type B and then back to A should lead to the same image) [216]

(continued)

Table 2 (continued)

Task	Technique	Description
3D point cloud processing (see Sect. 7.3)	PointNet++	Efficient NN architecture to process point cloud data [136]
	DGCNN	DGCNN learns to capture local geometric features of point clouds while still maintaining permutation invariance [196]
	pointCNN	Utilizes the advantageous CNN architecture to the irregular point cloud by learning a transformation on the input data to improve the final result [103]
	SO-Net	Self-organizing networks that learn permutation invariant features of the point cloud and are used both for classification, part segmentation, and generation [102]
Mesh processing (see Sect. 7.3)	MeshCNN	An NN architecture to process mesh data [58]
Graph neural networks (see Sect. 7.3)	GCN	Graph convolutional networks that rely on a simple layer-wise propagation rule [91]
	Graph-CNN	Graph-CNN defines filters as polynomials of functions of the graph adjacency matrix [173]
	Graph Attention Networks	Novel neural network architectures that operate on graph-structured data, leveraging masked self-attention layers to address the shortcomings of prior methods based on graph convolutions or their approximations [188]
Reinforcement Learning (see Sect. 8)	AlphaGo	The first neural network RL architecture to defeat a world champion in the difficult Go game [164]
	Continuous Control	An RL architecture to solve physical tasks. Using an actor-critic approach, this is a model-free algorithm based on the deterministic policy gradient that can operate over continuous action spaces [106]
	DDPG	Deep Deterministic Policy Gradient is an actor-critic, model-free algorithm that can operate over continuous action spaces [105]
	Deep Q-Learning	A deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning [105]. The input is raw pixels and the output is a value function estimating future rewards. A related model zoo is available here [138]
	SimPLe	Simulated Policy Learning is a complete model-based deep RL algorithm based on video prediction models [82]
AutoML (see Sect. 8)	DARTS	Differentiable architecture search for classification NN and RNN [110]. This framework decreases the search time and leads to good accuracy
	MNAS	Automated Mobile Neural Architecture Search (MNAS) approach to incorporate model latency into the main objective so that the search can identify a model that achieves a good trade-off between accuracy and latency [181]

Self-Supervised Learning (SSL) Online and incremental learning (see Sect. 7.3)	MoCo/SimCLR	Contrastive loss-based unsupervised representation learning for images [61, 17, 15, 16]
	EEIL	End-to-end incremental learning by combining distillation loss to remember old classes and cross-entropy for the new classes [12]
	Bias Correction	Use bias correction to improve incremental learning for large-scale datasets [199]
Domain adaptation (see Sect. 8)	DIRT-T	Decision-boundary Iterative Refinement Training with a Teacher model (DIRT-T) uses distillation and adversarial training to perform domain adaptation [161]
	CyCADA	Cycle-Consistent Adversarial Domain Adaptation is a domain adaptation-based approach for semantic segmentation that relies on the cycle consistency loss (like in Cycle-GAN) [70, 205]
Few-shot learning (see Sect. 8)	Graph Neural Networks	Graph neural representations for few-shot, semi-supervised, and active learning [49]
	MAML	Meta-learning-based few-shot learning [45]
	Prototypical Networks	Metric learning-based few-shot learning [169]
	Delta encoder	Generative network-based few-shot learning [155]
	RepMet	Metric learning based few-shot object detection [84]
Transfer learning (see Sect. 7.3)	Simultaneous Deep Transfer Across Domains and Tasks	A CNN architecture that effectively adapts to a new domain with limited or no labeled data per target category [185]
	Deep Transfer Learning with Joint Adaptation Networks	A joint adaptation network which learns a transfer network by aligning the joint distributions of multiple domain-specific layers across domains based on a joint maximum mean discrepancy criterion [115]
	Big Transfer (BiT)	A combination of heuristics that lead to the current state-of-the-art in transfer learning [92]
Object detection (see Sect. 7.1)	Faster-RCNN	A two-stage deep-learning-based object detection system that shares convolutional features between the region of interest (ROI) proposal part and the detection part [147, 195]
	YOLO	You only look once (YOLO) is an efficient one-stage network for object detection [142, 143, 144]
	FCOS	Fully Convolutional One-Stage Object Detection architecture to solve object detection in a per-pixel prediction fashion, analogue to semantic segmentation [184]
	Center-Net	An approach that detects each object as a triplet, rather than a pair, of keypoints, which improves both precision and recall [40]
Segmentation (see Sect. 7.1)	DeepLab	A system that re-purposes networks trained on image classification to the task of semantic segmentation by applying the "atrous convolution" with upsampled filters for dense feature extraction [14]
	Mask-RCNN	An instance segmentation network that extends Faster R-CNN by adding a branch for predicting an object mask in parallel to the existing branch for bounding box regression [62]

To summarize, being able to efficiently train deep neural networks has revolutionized almost every aspect of the modern day-to-day life. Examples span from bio-medical applications through computer graphics in movies and videos to international scale applications of big companies, such as Google, Amazon, Microsoft, Apple, and Facebook. Evidently, this theory is drawing much attention, and we believe there is still much to unravel, including exploring and understanding the NN's potential abilities and limitations.

The next chapters detail Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), generative models, and autoencoders. All are very important paradigms that are used in numerous applications (Table 2).

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3D point clouds. In: J. Dy, A. Krause (eds.) Proceedings of the 35th International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 80, pp. 40–49. PMLR, Stockholmsmässan, Stockholm Sweden (2018)
2. Atlason, H.E., AskellLove, Sigurdsson, S., Gudnason, V., Ellingsen, L.M.: Unsupervised brain lesion segmentation from MRI using a convolutional autoencoder. In: Medical Imaging 2019: Image Processing, vol. 10949, p. 109491H. International Society for Optics and Photonics (2019)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: 3rd International Conference on Learning Representations, ICLR (2015)
4. Bar, Y., Diamant, I., Wolf, L., Greenspan, H.: Deep learning with non-medical training used for chest pathology identification. In: Medical Imaging 2015: Computer-Aided Diagnosis, vol. 9414, pp. 215 – 221. International Society for Optics and Photonics, SPIE (2015)
5. Ben-Cohen, A., Diamant, I., Klang, E., Amitai, M., Greenspan, H.: Fully convolutional network for liver segmentation and lesions detection. In: Deep learning and data labeling for medical applications, pp. 77–85. Springer (2016)
6. Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: On the dangers of stochastic parrots: can language models be too big? In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 610–623 (2021). Association for Computing Machinery
7. Boscaini, D., Masci, J., Melzi, S., Bronstein, M.M., Castellani, U., Vandergheynst, P.: Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. Comput. Graph. Forum **34**, 13–23 (2015)
8. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (ICLR) (2019)
9. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: *Advances in Neural Information Processing Systems* (2020)
10. Caelles, S., Maninis, K.K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., Van Gool, L.: One-shot video object segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 221–230 (2017)
11. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 233–248 (2018)

12. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV, pp. 241–257 (2018)
13. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV), pp. 801–818 (2018)
14. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV (2018)
15. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020)
16. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.: Big self-supervised models are strong semi-supervised learners. arXiv preprint arXiv:2006.10029 (2020)
17. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
18. Chen, X., Xie, L., Wu, J., Tian, Q.: Progressive darts: Bridging the optimization gap for NAS in the wild. arXiv preprint arXiv:1912.10952 (2019)
19. Chen, Z., Tagliasacchi, A., Zhang, H.: BSP-Net: Generating compact meshes via binary space partitioning. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
20. Chen, Z., Zhang, J., Tao, D.: Progressive lidar adaptation for road detection. IEEE/CAA Journal of Automatica Sinica **6**(3), 693–702 (2019)
21. Cho, K., van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder–decoder approaches. In: Workshop on Syntax, Semantics and Structure in Statistical Translation, pp. 103–111. Association for Computational Linguistics (2014)
22. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (ELUs). CoRR (2015)
23. Cohen, G., Sapiro, G., Giryes, R.: DNN or K-NN: That is the generalize vs. memorize question. ArXiv **abs/1805.06822** (2018)
24. Cortes, C., Vapnik, V.: Support vector networks. Machine Learning **20**, 273–297 (1995)
25. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press (2000). https://doi.org/10.1017/CBO9780511801389
26. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: AutoAugment: Learning augmentation strategies from data. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
27. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: RandAugment: Practical automated data augmentation with a reduced search space. arXiv (2019)
28. Dahl, G.E., Sainath, T.N., Hinton, G.E.: Improving deep neural networks for LVCSR using rectified linear units and dropout. In: ICASSP, pp. 8609–8613. IEEE (2013)
29. Dauphin, Y.N., de Vries, H., Chung, J., Bengio, Y.: RMSProp and equilibrated adaptive learning rates for non-convex optimization. CoRR (2015)
30. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: International Conference on Neural Information Processing Systems, p. 3844–3852 (2016)
31. Dhariwal, P., Nichol, A.: Diffusion models beat GANs on image synthesis. CoRR (2021). https://arxiv.org/abs/2105.05233
32. Deng, J., Dong, W., Socher, R., jia Li, L., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. CVPR (2009)
33. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: ArcFace: Additive angular margin loss for deep face recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
34. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: ArcFace: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

35. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2018)
36. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (2019)
37. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv (2017)
38. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: E.P. Xing, T. Jebara (eds.) Proceedings of the 31st International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 32, pp. 647–655. PMLR, Beijing, China (2014)
39. Dovrat, O., Lang, I., Avidan, S.: Learning to sample. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
40. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: CenterNet: Keypoint triplets for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
41. Duchi, J., Hazan, E., yORAM Singer: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**, 2121–2159 (2011)
42. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. J. Mach. Learn. Res. **20**, 55:1–55:21 (2018)
43. van Engelen, J.E., Hoos, H.H.: A survey on semi-supervised learning. Machine Learning (2019). https://doi.org/10.1007/s10994-019-05855-6. https://doi.org/10.1007/s10994-019-05855-6
44. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (eds.) Advances in Neural Information Processing Systems 28, pp. 2962–2970. Curran Associates, Inc. (2015). http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf
45. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, p. 1126–1135. JMLR.org (2017)
46. Floridi, L., Chiriatti, M.: GPT-3: its nature, scope, limits, and consequences. Minds Mach. **30**, 681–694 (2020)
47. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. arXiv preprint arXiv:1409.7495 (2014)
48. Gao, C., Gu, D., Zhang, F., Yu, Y.: ReCoNet: Real-time coherent video style transfer network. In: Asian Conference on Computer Vision, pp. 637–653. Springer (2018)
49. Garcia, V., Bruna, J.: Few-shot learning with graph neural networks. In: 6th International Conference on Learning Representations, ICLR (2018)
50. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2414–2423 (2016)
51. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2414–2423 (2016)
52. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with LSTM. ICANN (1999)
53. Gibiansky, A., Arik, S., Diamos, G., Miller, J., Peng, K., Ping, W., Raiman, J., Zhou, Y.: Deep voice 2: Multi-speaker neural text-to-speech. In: Advances in neural information processing systems, pp. 2962–2970 (2017)
54. Goodfellow, I., Jean Pouget-Abadieand, M.M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 27, pp. 2672–2680. Curran Associates, Inc. (2014)

55. Greenspan, H., van Ginneken, B., Summers, R.M.: Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. CVPR **35**, 1153 – 1159 (2016)
56. Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., Wang, J.: Long text generation via adversarial training with leaked information. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
57. Haim, H., Elmalem, S., Giryes, R., Bronstein, A.M., Marom, E.: Depth estimation from a single image using deep learned phase coded mask. IEEE Transactions on Computational Imaging **4**(3), 298–310 (2018)
58. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: MeshCNN: A network with an edge. ACM Transactions on Graphics (TOG) **38**(4), 90 (2019)
59. Haris, M., Shakhnarovich, G., Ukita, N.: Deep back-projection networks for super-resolution. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
60. Hayashi, T., Yamamoto, R., Inoue, K., Yoshimura, T., Watanabe, S., Toda, T., Takeda, K., Zhang, Y., Tan, X.: ESPnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7654–7658 (2020)
61. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR (2020)
62. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. IEEE International Conference on Computer Vision (ICCV) pp. 2980–2988 (2017)
63. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
64. Hertz, A., Hanocka, R., Giryes, R., Cohen-Or, D.: Deep geometric texture synthesis. ACM Trans. Graph. (2020)
65. Hertz, A., Hanocka, R., Giryes, R., Cohen-Or, D.: PointGMM: A neural GMM network for point clouds. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12051–12060 (2020)
66. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006). https://doi.org/10.1162/neco.2006.18.7.1527. URL http://dx.doi.org/10.1162/neco.2006.18.7.1527
67. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, H. Lin (eds.) *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851. Curran Associates, Inc. (2020)
68. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
69. Hoffman, J., Tzeng, E., Park, T., Zhu, J.Y., Isola, P., Saenko, K., Efros, A., Darrell, T.: CyCADA: Cycle-consistent adversarial domain adaptation. In: J. Dy, A. Krause (eds.) Proceedings of the 35th International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 80, pp. 1989–1998. PMLR, Stockholmsmässan, Stockholm Sweden (2018)
70. Hoffman, J., Tzeng, E., Park, T., Zhu, J.Y., Isola, P., Saenko, K., Efros, A., Darrell, T.: CyCADA: Cycle-consistent adversarial domain adaptation. In: J. Dy, A. Krause (eds.) Proceedings of Machine Learning Research, vol. 80, pp. 1989–1998. PMLR, Stockholmsmässan, Stockholm Sweden (2018)
71. Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: RandLA-Net: Efficient semantic segmentation of large-scale point clouds. arXiv preprint arXiv:1911.11236 (2019)
72. Huang, G., Liu, Z., Maaten, L.V.D., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700–4708 (2017)
73. Hubel, D.H., Wiesel, T.N.: Receptive fields of single neurons in the cat's striate cortex. Journal of Physiology **148**, 574–591 (1959)

74. Hussein, S.A., Tirer, T., Giryes, R.: Correction filter for single image super-resolution: Robustifying off-the-shelf deep super-resolvers. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1425–1434 (2020)
75. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning, vol. 37, pp. 448–456 (2015)
76. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 5967–5976 (2017)
77. Jain, L.C., Medsker, L.R.: Recurrent Neural Networks: Design and Applications, 1st edn. CRC Press, Inc., Boca Raton, FL, USA (1999)
78. Jakubovitz, D., Giryes, R., Rodrigues, M.R.D.: Generalization Error in Deep Learning, pp. 153–193. Springer International Publishing, Cham (2019)
79. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision, pp. 694–711. Springer (2016)
80. Kadlec, R., Schmid, M., Bajgar, O., Kleindienst, J.: Text understanding with the attention sum reader network. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 908–918. Association for Computational Linguistics, Berlin, Germany (2016). https://doi.org/10.18653/v1/P16-1086
81. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of artificial intelligence research **4**, 237–285 (1996)
82. Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Sepassi, R., Tucker, G., Michalewski, H.: Model-based reinforcement learning for Atari. ArXiv **abs/1903.00374** (2020)
83. Kalogerakis, E., Averkiou, M., Maji, S., Chaudhuri, S.: 3d shape segmentation with projective convolutional networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 6630–6639 (2016)
84. Karlinsky, L., Shtok, J., Harary, S., Schwartz, E., Aides, A., Feris, R., Giryes, R., Bronstein, A.M.: RepMet: Representative-based metric learning for classification and few-shot object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
85. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: International Conference on Learning Representations (2018). https://openreview.net/forum?id=Hk99zCeAb
86. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4396–4405 (2019)
87. Kemker, R., McClure, M., Abitino, A., Hayes, T.L., Kanan, C.: Measuring catastrophic forgetting in neural networks. In: Thirty-second AAAI conference on artificial intelligence (2018)
88. Keskar, N.S., Socher, R.: Improving generalization performance by switching from Adam to SGD. arXiv preprint arXiv:1712.07628 (2017)
89. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR (2014)
90. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
91. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
92. Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., Houlsby, N.: Big transfer (bit): General visual representation learning. In: ECCV (2020)
93. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012)
94. Krogh, A., Hertz, J.A.: A simple weight decay can improve generalization. In: J.E. Moody, S.J. Hanson, R.P. Lippmann (eds.) Advances in Neural Information Processing Systems 4,

pp. 950–957. Morgan-Kaufmann (1992). http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization.pdf
95. Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.: Deep convolutional inverse graphics network. In: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (eds.) Advances in Neural Information Processing Systems 28, pp. 2539–2547. Curran Associates, Inc. (2015). http://papers.nips.cc/paper/5851-deep-convolutional-inverse-graphics-network.pdf
96. Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., Matas, J.: DeblurGan: Blind motion deblurring using conditional adversarial networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8183–8192 (2018)
97. Kwon, D., Kim, H., Kim, J., Suh, S.C., Kim, I., Kim, K.J.: A survey of deep learning-based network anomaly detection. Cluster Computing **22**(1), 949–961 (2019). https://doi.org/10.1007/s10586-017-1117-8
98. Lample, G., Conneau, A., Denoyer, L., Ranzato, M.: Unsupervised machine translation using monolingual corpora only. arXiv preprint arXiv:1711.00043 (2017)
99. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989). https://doi.org/10.1162/neco.1989.1.4.541. URL http://dx.doi.org/10.1162/neco.1989.1.4.541
100. LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., Jackel, L.D.: Hand-written digit recognition with a back-propagation network. NIPS (1990)
101. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, pp. 2278–2324 (1998)
102. Li, J., Chen, B.M., Lee, G.H.: So-net: Self-organizing network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
103. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: Convolution on X-transformed points. In: NeurIPS (2018)
104. Li, Y., Pirk, S., Su, H., Qi, C.R., Guibas, L.J.: FPNN: Field probing neural networks for 3d data. In: D.D. Lee, M. Sugiyama, U.V. Luxburg, I. Guyon, R. Garnett (eds.) Advances in Neural Information Processing Systems 29, pp. 307–315. Curran Associates, Inc. (2016). http://papers.nips.cc/paper/6416-fpnn-field-probing-neural-networks-for-3d-data.pdf
105. Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: ICLR (2016)
106. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: Y. Bengio, Y. LeCun (eds.) ICLR (2016)
107. Lim, S., Kim, I., Kim, T., Kim, C., Kim, S.: Fast AutoAugment. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
108. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988 (2017)
109. Liu, G., Reda, F.A., and Ting-Chun Shih, K.J.S., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions. In: The European Conference on Computer Vision (ECCV) (2018)
110. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable architecture search. In: International Conference on Learning Representations (2019)
111. Liu, H.T.D., Kim, V.G., Chaudhuri, S., Aigerman, N., Jacobson, A.: Neural subdivision. ACM Trans. Graph. **39**(4) (2020)
112. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: ECCV (2016)
113. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L.: SphereFace: Deep hypersphere embedding for face recognition. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 6738–6746 (2017)

114. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized Bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
115. Long, M., Zhu, H., Wang, J., Jordan, M.I.: Deep transfer learning with joint adaptation networks. In: International conference on machine learning, pp. 2208–2217. PMLR (2017)
116. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2017)
117. Ma, W.C., Wang, S., Hu, R., Xiong, Y., Urtasun, R.: Deep rigid instance scene flow. In: CVPR (2019)
118. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics **5**(4), 115–133 (1943). https://doi.org/10.1007/BF02478259
119. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 26, pp. 3111–3119. Curran Associates, Inc. (2013). http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
120. Minsky, M., Papert, S.: Perceptrons: An Introduction to Computational Geometry. MIT Press, Cambridge, MA, USA (1969)
121. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: International Conference on Learning Representations (2018). https://openreview.net/forum?id=B1QRgziT-
122. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing Atari with deep reinforcement learning. ArXiv **abs/1312.5602** (2013)
123. Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model CNNs. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 5425–5434 (2017). https://doi.org/10.1109/CVPR.2017.576
124. Nah, S., Kim, T.H., Lee, K.M.: Deep multi-scale convolutional neural network for dynamic scene deblurring. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3883–3891 (2017)
125. Nesterov, Y.E.: A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In: Dokl. akad. nauk Sssr, vol. 269, pp. 543–547 (1983)
126. Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, pp. 2014–2023. JMLR.org (2016). http://dl.acm.org/citation.cfm?id=3045390.3045603
127. Noy, A., Nayman, N., Ridnik, T., Zamir, N., Doveh, S., Friedman, I., Giryes, R., Zelnik-Manor, L.: Asap: Architecture search, anneal and prune. arXiv preprint arXiv:1904.04123 (2019)
128. Ongie, G., Willett, R., Soudry, D., Srebro, N.: A function space view of bounded norm infinite width re{lu} nets: The multivariate case. In: International Conference on Learning Representations (ICLR) (2020)
129. van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. In: Arxiv (2016). https://arxiv.org/abs/1609.03499
130. Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q.: Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks **22**(2), 199–210 (2010)
131. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: Learning continuous signed distance functions for shape representation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
132. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: S. Dasgupta, D. McAllester (eds.) Proceedings of the 30th International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 28, pp. 1310–1318. PMLR, Atlanta, Georgia, USA (2013)

133. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proc. of NAACL (2018)
134. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations (2018). http://arxiv.org/abs/1802.05365
135. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3d classification and segmentation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 77–85 (2016)
136. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017)
137. Qian, N.: On the momentum term in gradient descent learning algorithms. Neural Networks **12**(1), 145–151 (1999)
138. Quan, J., Ostrovski, G.: DQN Zoo: Reference implementations of DQN-based agents (2020). http://github.com/deepmind/dqn_zoo
139. Radford, A., Sutskever, I.: Improving language understanding by generative pre-training. In: arxiv (2018)
140. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. Technical report, OpenAI (2019)
141. Reddi, S.J., Kale, S., Kumar, S.: On the convergence of Adam and beyond. In: International Conference on Learning Representations (ICLR) (2018)
142. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 779–788 (2015)
143. Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 6517–6525 (2016)
144. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. ArXiv **abs/1804.02767** (2018)
145. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, p. 1060–1069. JMLR.org (2016)
146. Remez, T., Litany, O., Giryes, R., Bronstein, A.M.: Class-aware fully convolutional Gaussian and Poison denoising. IEEE Transactions on Image Processing **27**(11), 5707–5722 (2018)
147. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (eds.) Advances in Neural Information Processing Systems 28, pp. 91–99. Curran Associates, Inc. (2015)
148. Ronneberger, O., P.Fischer, Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI), *LNCS*, vol. 9351, pp. 234–241. Springer (2015)
149. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review pp. 65–386 (1958)
150. Ruck, D.W., Rogers, S.K.: Feature Selection Using a Multilayer Perceptron. Journal of Neural Network Computing **2**(July 1993), 40–48 (1990)
151. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Representations by Back-propagating Errors. Nature **323**(6088), 533–536 (1986). https://doi.org/10.1038/323533a0
152. Safran, I., Eldan, R., Shamir, O.: Depth separations in neural networks: What is actually being separated? In: Conference on Learning Theory (COLT), pp. 2664–2666. PMLR (2019)
153. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. CVPR pp. 815–823 (2015)
154. Schwartz, E., Giryes, R., Bronstein, A.M.: DeepISP: Toward learning an end-to-end image processing pipeline. IEEE Transactions on Image Processing **28**(2), 912–923 (2019). https://doi.org/10.1109/TIP.2018.2872858

155. Schwartz, E., Karlinsky, L., Shtok, J., Harary, S., Marder, M., Kumar, A., Feris, R., Giryes, R., Bronstein, A.: Delta-encoder: an effective sample synthesis method for few-shot object recognition. In: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (eds.) Advances in Neural Information Processing Systems 31, pp. 2845–2855. Curran Associates, Inc. (2018)
156. Shaham, T.R., Dekel, T., Michaeli, T.: SinGAN: Learning a generative model from a single natural image. In: The IEEE International Conference on Computer Vision (ICCV) (2019)
157. Shen, J., Pang, R., Weiss, R.J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., Saurous, R.A., Agiomvrgiannakis, Y., Wu, Y.: Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4779–4783 (2018)
158. Shiloh, L., Eyal, A., Giryes, R.: Efficient processing of distributed acoustic sensing data using a deep learning approach. J. Lightwave Technol. **37**(18), 4755–4762 (2019)
159. Shocher, A., Bagon, S., Isola, P., Irani, M.: InGAN: Capturing and retargeting the "DNA" of a natural image. In: The IEEE International Conference on Computer Vision (ICCV) (2019)
160. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. Journal of Big Data **6**(1), 60 (2019). https://doi.org/10.1186/s40537-019-0197-0
161. Shu, R., Bui, H., Narui, H., Ermon, S.: A DIRT-t approach to unsupervised domain adaptation. In: International Conference on Learning Representations (2018). https://openreview.net/forum?id=H1q-TM-AW
162. Shu, R., Bui, H.H., Narui, H., Ermon, S.: A DIRT-T approach to unsupervised domain adaptation. In: 6th International Conference on Learning Representations, ICLR (2018)
163. Shwartz-Ziv, R., Tishby, N.: Opening the black box of deep neural networks via information. ArXiv **abs/1703.00810** (2017)
164. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. Nature **550**, 354–372 (2017)
165. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2015)
166. Singh, S., Krishnan, S.: Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. arXiv (2019)
167. Smith, L.N.: Cyclical learning rates for training neural networks. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 464–472 (2017)
168. Smith, L.N.: A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. ArXiv **abs/1803.09820** (2018)
169. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems (2017)
170. Sønderby, C.K., Raiko, T., Maaløe, L., Sønderby, S.K., Winther, O.: Ladder variational autoencoders. In: Advances in neural information processing systems, pp. 3738–3746 (2016)
171. Sotelo, J., Mehri, S., Kumar, K., Santos, J.F., Kastner, K., Courville, A.C., Bengio, Y.: Char2wav: End-to-end speech synthesis. In: ICLR (2017)
172. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
173. Such, F.P., Sah, S., Domínguez, M., Pillai, S., Zhang, C., Michael, A., Cahill, N.D., Ptucha, R.W.: Robust spatial filtering with graph convolutional neural networks. IEEE Journal of Selected Topics in Signal Processing **11**, 884–896 (2017)
174. Sun, Q., Liu, Y., Chua, T.S., Schiele, B.: Meta-transfer learning for few-shot learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
175. Sun, R.: Optimization for deep learning: theory and algorithms. arXiv preprint arXiv:1912.08957 (2019)
176. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1199–1208 (2018)

177. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112. Curran Associates, Inc. (2014)
178. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
179. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: DeepFace: Closing the gap to human-level performance in face verification. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
180. Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C.: A survey on deep transfer learning. In: V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, I. Maglogiannis (eds.) Artificial Neural Networks and Machine Learning – ICANN 2018, pp. 270–279. Springer International Publishing, Cham (2018)
181. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: MnasNet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2820–2828 (2019)
182. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR, Long Beach, California, USA (2019)
183. Tetko, I.V., Livingstone, D.J., Luik, A.I.: Neural network studies. 1. comparison of overfitting and overtraining. Journal of chemical information and computer sciences **35**(5), 826–833 (1995)
184. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), ICCV '19. IEEE Computer Society (2019)
185. Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Simultaneous deep transfer across domains and tasks. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 4068–4076 (2015)
186. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2962–2971 (2017)
187. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems 30, pp. 5998–6008. Curran Associates, Inc. (2017). http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
188. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. ArXiv **abs/1710.10903** (2018)
189. Vidal, R., Bruna, J., Giryes, R., Soatto, S.: Mathematics of deep learning (2017)
190. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning, pp. 1096–1103 (2008)
191. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
192. Wang, H., Wang, Y., Zhou, Z., Ji, X., Li, Z., Gong, D., Zhou, J., Liu, W.: Cosface: Large margin cosine loss for deep face recognition. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018)
193. Wang, H., Wang, Y., Zhou, Z., Ji, X., Li, Z., Gong, D., Zhou, J., Liu, W.: Cosface: Large margin cosine loss for deep face recognition. 2018 IEEE/CVF Conference on Computer Vision and Pa Recognition pp. 5265–5274 (2018)
194. Wang, X., Chan, K.C., Yu, K., Dong, C., Loy, C.C.: EDVR: Video restoration with enhanced deformable convolutional networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2019)

195. Wang, X., Shrivastava, A., Gupta, A.: A-Fast-RCNN: Hard positive generation via adversary for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2606–2615 (2017)
196. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Transactions on Graphics (TOG) (2019)
197. Wang, Y., Yao, Q.: Generalizing from a few examples: A survey on few-shot learning. ArXiv (2019)
198. Wilson, G., Cook, D.J.: A survey of unsupervised deep domain adaptation. In: arxiv (2018)
199. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 374–382 (2019)
200. Wu, Y., He, K.: Group normalization. In: The European Conference on Computer Vision (ECCV) (2018)
201. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1912–1920 (2014)
202. Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network. ArXiv (2015)
203. Yang, C., Lu, X., Lu, Z., Shechtman, E., Wang, O., Li, H.: High-resolution image inpainting using multi-scale neural patch synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6721–6729 (2017)
204. Yang, W., Zhang, X., Tian, Y., Wang, W., Xue, J., Liao, Q.: Deep learning for single image super-resolution: A brief review. IEEE Transactions on Multimedia **21**(12), 3106–3121 (2019). https://doi.org/10.1109/TMM.2019.2919431
205. Yang, Y., Soatto, S.: FDA: Fourier domain adaptation for semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4085–4095 (2020)
206. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V.: XLNet: Generalized autoregressive pretraining for language understanding (2019)
207. Yang, Z., Dai, Z., Yang, Y., Carbonell, J.G., Salakhutdinov, R., Le, Q.V.: XLNet: Generalized autoregressive pretraining for language understanding. CoRR **abs/1906.08237** (2019). http://arxiv.org/abs/1906.08237
208. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. In: The IEEE International Conference on Computer Vision (ICCV) (2019)
209. Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H., Shao, L.: CycleISP: Real image restoration via improved data synthesis. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2693–2702 (2020)
210. Zeiler, M.D.: ADADELTA: An adaptive learning rate method. ArXiv **abs/1212.5701** (2012)
211. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations (2018). https://openreview.net/forum?id=r1Ddp1-Rb
212. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. IEEE Transactions on Image Processing **26**(7), 3142–3155 (2017)
213. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image super-resolution using very deep residual channel attention networks. In: ECCV (2018)
214. Zhao, H., Gallo, O., Frosio, I., Kautz, J.: Loss functions for image restoration with neural networks. IEEE Transactions on Computational Imaging **3**, 47–57 (2017)
215. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 2242–2251 (2017)
216. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision, pp. 2223–2232 (2017)

Graph Embedding

Palash Goyal

1 Introduction

Graphs are ubiquitous. Most things in the real world interact with each other forming a large variety of graphs. For example, people interact on social media by befriending and commenting and liking posts forming a large social graph. Similarly, atoms and molecules interact to form biological interaction graph. In language, words interact to form sentences which in turn interact to form paragraphs. These interactions can also be modeled as word co-occurrence graphs or sentence graphs. Thus, analyzing and understanding graphs is crucial to getting insights into the physical world around us. Their application to a vast variety of domains has led to tremendous efforts in research for the field. Originally, the idea was introduced by Leonhard Euler in eighteenth century to define walks in a city connected by bridges. Today, graph algorithms are used to navigate around the world, recommend friends and movies, and even to predict formation of new compounds.

Another branch of science which has grown over the past few decades is artificial intelligence and machine learning. As we have studied in the previous chapters, methods such as support vector machines and neural networks have let us model dependencies between features available to predict the outcome. They have been used to predict weather trends, estimate risk of cardiovascular death, and automatically translate text in different languages. The impact on business applications of these machine learning methods is massive, since they affect so many different areas like machine translation, healthcare diagnostics, chat-bot behavior, warehouse inventory management, automated email responses, facial recognition, and customer review analysis, just to name a few. Driven by these applications, the

P. Goyal (✉)
University of Southern California, Los Angeles, CA, USA
e-mail: palashgo@usc.edu

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_15

machine learning models have advanced to capture complex patterns of information from data to learn dependencies with the output variable.

Graph embedding a.k.a. network representation learning lies at the intersection of graph analysis and machine learning. Graphs are typically represented as a set of nodes/vertices and edges. Each node is defined by the presence of edges with its neighbors and absence with the rest of the nodes. Thus, they are very high dimensional. However, most of the machine learning methods require an absolute definition of a data point instead of the relative definition as provided by the graphs. This gave rise to graph embedding which aims to learn a low-dimensional absolute representation of each node from the graph.

Learning such low-dimensional representation can be used for many graph tasks including link prediction, node classification and regression, and graph visualization. Link prediction refers to the task of predicting missing links or links that are likely to occur in the future. Node classification aims at determining the label of nodes (a.k.a. vertices) based on other labeled nodes and the topology of the network. Clustering is used to find subsets of similar nodes and group them together; finally, visualization helps in providing insights into the structure of the network.

We can categorize the graph embedding problems into three categories: (i) static or snapshot graph embedding, (ii) dynamic or temporal graph embedding, and (iii) attributed graph embedding. Static methods look at a single snapshot of a graph and learn the representation of each node for that snapshot. Dynamic methods extend this to temporal graphs and learn embeddings across time. Attributed methods extend traditional graph embedding methods to work for graphs with node and edge attributes.

The rest of the chapter is organized as follows: Sect. 2 defines the problem statement and the notations, Sect. 3 introduces methods for static graph embedding, Sect. 4 draws insights into the problem of dynamic graph embedding, and Sect. 5 explains approaches recently introduced for attributed graph embedding.

2 Definitions

We now define some of the most common terms typically used in the literature of graph embedding [1, 2].

Definition 1 (Graph) A graph $G(V, E)$ is a collection of $V = \{v_1, \cdots, v_n\}$ vertices (a.k.a. nodes) and $E = \{e_{ij}\}_{i,j=1}^{n}$ edges. The adjacency matrix S of graph G contains non-negative weights associated with each edge: $s_{ij} \geq 0$. If v_i and v_j are not connected to each other, then $s_{ij} = 0$. For undirected weighted graphs, $s_{ij} = s_{ji} \ \forall i, j \in [n]$.

The edge weight s_{ij} is generally treated as a measure of similarity between the nodes v_i and v_j. The higher the edge weight, the more similar the two nodes are expected to be.

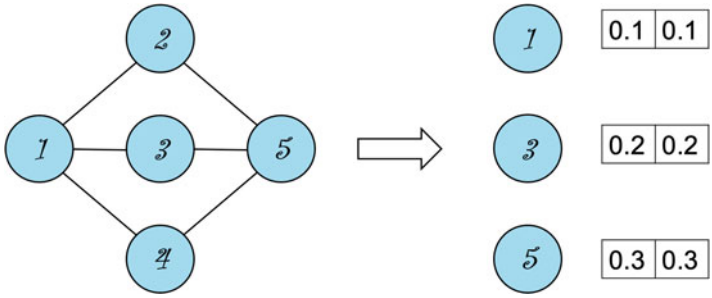


Fig. 1 Example illustrating graph embedding. Nodes 1 and 3 are first-order neighbors and should be close in the embedding space. Nodes 1 and 5 are second-order neighbors and should be relatively farther

Definition 2 (First-Order Proximity) Edge weights s_{ij} are also called first-order proximities between nodes v_i and v_j, since they are the first and foremost measures of similarity between two nodes. In Fig. 1, nodes 1, 2, 3, 4, and 5 form the set of nodes and all edges have equal weights. Nodes 1 and 2 are connected and are thus first-order neighbors.

We can similarly define higher order proximities between nodes. For instance, see the following definition:

Definition 3 (Second-Order Proximity) The second-order proximity between a pair of nodes describes the proximity of the pair's neighborhood structure. Let $\boldsymbol{s_i} = [s_{i1}, \cdots, s_{in}]$ denote the first-order proximity between v_i and other nodes. Then, second-order proximity between v_i and v_j is determined by the similarity of $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$.

The second-order proximity compares the neighborhood of two nodes and treats them as similar if they have a similar neighborhood. In Fig. 2, nodes 1 and 5 are second-order neighbors. It is possible to define higher order proximities using other metrics, e.g., Katz Index [30], Rooted PageRank [33], Common Neighbors [32], Adamic-Adar [31], etc. Next, we define a graph embedding:

Definition 4 (Graph Embedding) Given a graph $G = (V, E)$, a graph embedding is a mapping $f : v_i \rightarrow \boldsymbol{y_i} \in \mathbb{R}^d \ \forall i \in [n]$ such that $d \ll |V|$ and the function f preserves some proximity measure defined on graph G.

An embedding therefore maps each node to a low-dimensional feature vector and tries to preserve the connection strengths between vertices. For instance, an embedding preserving first-order proximity might be obtained by minimizing $\sum_{i,j} s_{ij} \|\boldsymbol{y_i} - \boldsymbol{y_j}\|_2^2$. Let two node pairs (v_i, v_j) and (v_i, v_k) be associated with connection strengths such that $s_{ij} > s_{ik}$. In this case, v_i and v_j will be mapped to points in the embedding space that will be closer to each other than the mapping of v_i and v_k. In Fig. 2, nodes 1 and 3 should be closer in embedding space than 4 and 2 as 1 and 3 have a direct relation between them.

Definition 5 (Dynamic Graph Embedding) Given an evolution of graph G, $\mathcal{G}$, a dynamic graph embedding aims to represent each node v in a series of low-dimensional vector space $y_{v_1}, \ldots y_{v_t}$, where y_{v_t} is the embedding of node v at time t, by learning mappings $f_t : \{V_1, \ldots, V_t, E_1, \ldots E_t\} \rightarrow \mathbb{R}^d$ and $y_{v_i} = f_i(v_1, \ldots, v_i, E_1, \ldots E_i)$ such that y_{v_i} can capture temporal patterns required to predict $y_{v_{i+1}}$.

A dynamic graph embedding thus works for a series of graphs. It can be used to update embeddings of graphs over time or capture temporal patterns of graph evolution to generate a temporally aware embedding. As most data from real world often contain attributes on nodes and edges, there are several works on attributed graph embedding as defined below:

Definition 6 (Attributed Graph Embedding) Given a graph $G = (V, E)$ and associated node and edge attributes V^a and E^a, respectively, attributed graph embedding aims to represent each node u in a low-dimensional vector space $\boldsymbol{y_u}$ by learning a mapping $f : \{V, V^a, E^a\} \rightarrow \mathbb{R}^d$, namely $\boldsymbol{y_v} = f(v, V^a, E^a)\ \forall v \in V$.

An attributed graph embedding method aims to preserve both the structural properties of the graph and the additional attribute information available.

Several methods have been proposed preserving various properties and attributes of the graph. This increase in the methods has led researchers to extend the concept of ensemble learning to graph embedding domain:

Definition 7 (Ensemble Graph Embedding) Given a set of embedding methods $\{m_1, \ldots, m_k\}$ with corresponding embeddings for a graph G as $\{X^{m_1}, \ldots, X^{m_k}\}$ and errors $\{\epsilon_1, \ldots, \epsilon_k\}$ on a graph task $\mathcal{T}$, a graph ensemble learning approach aims to learn an embedding X^m with error ϵ such that $\epsilon < min(\epsilon_1, \ldots, \epsilon_k)$.

An ensemble approach leverages the accuracy and diversity of the individual models to make more accurate predictions.

3 Static Graph Embedding Methods

Static graph embedding methods aim to learn representations of nodes for a snapshot of a graph. The snapshot may be cumulative over time or may represent a single timestamp. Research in this field is driven by the following questions:

? Questions

1. Which properties of the input graph are useful for downstream tasks such as node classification and link prediction?
2. How can we capture the underlying manifold of the graph?
3. How can we develop methods which can scale for large graphs?
4. How much space is really required to store a graph?

Graph embedding methods quantify the above questions and define ways to answer them. They give us a deeper understanding of the information contained in the graph and how it can be utilized for downstream tasks. We will now explain how different methods attempt to answer the questions.

3.1 Graph Properties Preserved

Graph analysis over the decades has led to a discovery of several properties real-world graphs may have. Social networks often have people connected in densely connected communities with sparse connections between the communities. Web graphs have been shown to have structural properties with websites acting as hubs and authorities of information content. Biology networks have a mixture of the above properties.

Consider a social network graph in Fig. 2. It is composed of several properties. Persons a to i form a community closely connected to one another. Tom and Nancy also have a special role of connecting the two communities. In contrast, Peter is not a part of either of the communities but serves as a link between them.

To predict missing links or possible future links for this graph, it is important to capture both the community structure and the structural roles of the people in the graph. Based on the properties preserved in the approach, there are two categories of graph embedding methods: (i) community preserving and (ii) structure preserving.

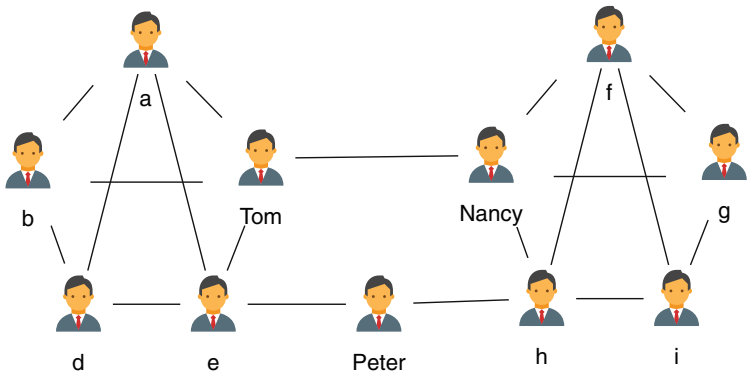


Fig. 2 Example illustrating a social graph with multiple properties. Persons a to i form two communities. Tom and Nancy have a special role of being in individual community but connecting them. Peter is not a part of either of the communities but is a link between them

3.1.1 Community Preserving Graph Embedding Methods

Community preserving methods aim to capture the distances in the input graph in the embedding space. Within this category, the methods vary on the level of distance captures. For example, Graph Factorization [3] and Laplacian Eigenmaps [4] preserve shorter distances (i.e., low-order proximity) in the graph, whereas more recent methods such as Higher Order Proximity Embedding (HOPE) [5] and GraRep [6] capture longer distances (i.e., high order proximity).

Here are some examples of such methods:

1. **Laplacian Eigenmaps**: Laplacian Eigenmaps [4] keeps the embedding of two nodes close when the weight W_{ij} is high. Specifically, they minimize the objective function $\phi(Y) = \frac{1}{2}\sum_{i,j}(Y_i - Y_j)^2 W_{ij} = Y^T L Y$, where L is the Laplacian of graph G. The solution to this can be obtained by taking the eigenvectors corresponding to the d smallest eigenvalues of the normalized Laplacian, $L_{norm} = D^{-1/2} L D^{-1/2}$.
2. **Graph Factorization**: Graph Factorization [3] was the first method to obtain a graph embedding in $O(|E|)$ time. To obtain the embedding, GF factorizes the adjacency matrix of the graph, minimizing the following loss function:

$$\phi(Y, \lambda) = \frac{1}{2} \sum_{(i,j) \in E} (W_{ij} - < Y_i, Y_j >)^2 + \frac{\lambda}{2} \sum_i \|Y_i\|^2,$$

   where λ is a regularization coefficient.
3. **HOPE**: HOPE [5] preserves higher order proximity by minimizing $\|S - Y_s Y_t^T\|_F^2$, where S is the similarity matrix. The authors experimented with different similarity measures, including Katz Index, Rooted Page Rank, Common Neighbors, and Adamic-Adar score. They represented each similarity measure as $S = M_g^{-1} M_l$, where both M_g and M_l are sparse. This enables HOPE to use generalized Singular Value Decomposition (SVD) to obtain the embedding efficiently.

3.1.2 Structure Preserving Graph Embedding Methods

Structure preserving methods aim to understand the structural similarity between nodes and capture role of each node. *node2vec* [7] uses a mixture of breadth-first and depth-first search for this. Deep learning methods such as Structural Deep Network Embedding (SDNE) [8] and Deep Network Graph Representation (DNGR) [6] use deep autoencoders to preserve distance and structure. Here are some examples of such methods:

1. **Structural Deep Network Embedding (SDNE)**: Wang et al. [8] proposed to use deep autoencoders to preserve the first- and second-order network proximities. They achieve this by jointly optimizing the two proximities. The

approach uses highly nonlinear functions to obtain the embedding. The model consists of two parts: unsupervised and supervised. The former consists of an autoencoder aiming at finding an embedding for a node which can reconstruct its neighborhood. The latter is based on Laplacian Eigenmaps [4] which apply a penalty when similar vertices are mapped far from each other in the embedding space.

2. **node2vec**: node2vec [7] preserves higher order proximity between nodes by maximizing the probability of observing the last k nodes and the next k nodes in the random walk centered at v_i, i.e., maximizing $\log Pr(v_{i-k}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+k}|Y_i)$, where $2k + 1$ is the length of the random walk. It employs biased random walks that provide a trade-off between breadth-first (BFS) and depth-first (DFS) graph searches.

3.2 Manifold Learning

An alternate way of looking at graph embedding is that the low-dimensional representation is learning the underlying manifold the graph lies in. The manifold can be simple and linear or complex depending on the connections between nodes. Methods vary in their definition of the function approximating the underlying manifold. Below, we mention three broad classes of methods based on this:

1. **Matrix Factorization**: They represent graph as a similarity matrix and decompose it to get the embedding. Graph Factorization and HOPE use adjacency matrix and higher order proximity matrix for this.
2. **Deep Learning**: They use multiple nonlinear layers to capture the underlying manifold of the interactions between nodes. SDNE, DNGR, and VGAE [9] are examples of these methods. Success in this has given rise to the field of graph neural networks [34], which uses message passing between nodes to capture the relations in graphs.
3. **Hyperbolic Embedding**: Instead of representing the nodes in a Euclidean space, they use a hyperbolic space such as Poincare disk. These methods [10, 11] claim that hyperbolic space is optimal and showcase it by illustrating performance in downstream task.

3.3 Model Scalability

Real-world networks lie on a broad spectrum of graph size. Biological networks are typically small, whereas web networks span the entire Internet and are in the order of billions of nodes. Thus, understanding the time complexity of methods is crucial for any application. Faster methods typically use simpler function approximations, whereas highly nonlinear models are usually very computationally expensive. Based

on the time taken to learn the embedding in terms of the number of edges in the graph, methods can be put into the following categories:

1. **Sub-linear Methods**: These methods use $O(|V|)$ time. Random walk-based methods (e.g., DeepWalk, *node2vec*) that take a random walk of constant length from each node fall in this category.
2. **Linear Methods**: Methods that traverse each edge a constant number of times and take $O(|E|)$ time fall in this category. Sparse matrix factorization-based approaches typically fall in this category.
3. **Quadratic Methods**: These methods take quadratic time even for sparse graphs. Matrix factorization-based methods that compute a dense similarity matrix from the adjacency matrix and deep learning methods fall in this category.

Furthermore, there are some methods which aim to reduce the memory footprint of the representation by embedding the nodes in a discrete space. Shen et al. [12] use hamming distance to approximate the ground truth distance and learn the binary code for each node.

4 Dynamic Graph Embedding Methods

Static graph embedding methods are useful for understanding the properties of any graph and get an estimate of the node characteristics. However, most real-world graphs evolve over time. Friendship graphs evolve by people changing communities, severing old connections, and forming new ones. Similarly, router networks evolve by addition of new routers and removal of faulty ones. This leads to the following questions:

? Questions

1. How can we efficiently update graph embedding as the connections evolve over time?
2. How can we capture temporal patterns in dynamic graphs?

Over the past 5 years, understanding dynamics of graphs and incorporating them in the study of graph embedding has been studied more extensively.

4.1 Updating Graph Embedding

Rate of evolution of node and edges in a graph varies with the domain. Router networks and biology networks are relatively more stable whereas social networks and information networks grow rapidly. Furthermore, the evolution is local in some cases, e.g., a person changing a job in a job network, and global in other cases, e.g., spread of epidemic. Recalculating the embedding of a graph after short periods of local changes is not efficient. On the other hand, keeping the same embedding although the network has changed makes the embedding less informative.

Figure 3 illustrates a job network before and after Tom changes his job. We observe that although the local network around Tom has changed, the rest of the graph remains fairly constant. Recommending friends from his previous job is less significant given the job change. Thus, it is important to modify Tom's representation to denote that.

Several methods have been proposed in the recent years to efficiently update graph embeddings for dynamic graphs. A pioneer work in this field is by Zhu et al. [13]. The method extends Graph Factorization for static graphs by introducing a temporal regularizer $1 - y_{v_t} y_{v_{t-1}}^T$ for a node v at time steps t and $t - 1$ penalizing a drastic change in the dot product distance between the embedding of a node at consecutive time steps. TIMERS [14] use a similar temporal regularization for Singular Value Decomposition of a graph.

DynGEM [15] proposed a deep neural network-based approach without the regularization by initialization the neural network weights of the model for time step t with $t - 1$ and incrementally updating the weights. They showed superior performance to the previous methods. In addition, they provided a way to dynamically adapt the size of neural network to increasing information contained in the graph.

DynamicTriad [16] relaxes the temporal smoothness assumption and captures patterns spanning two time steps. DyLink2vec [17] transforms the problem of learning temporal changes to learning embeddings of links and using time series to capture the patterns. Other methods using similar approaches [18, 19] extending static methods such as LINE and graph neural networks have been proposed.

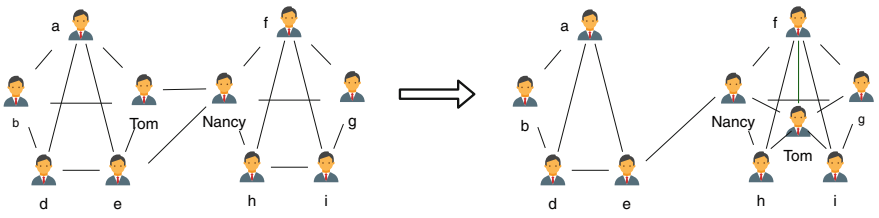


Fig. 3 Example illustrating a job network before and after Tom changes his job. The local network around Tom has changed, but the rest of the graph remains mostly unchanged

4.2 Capturing Temporal Patterns

Certain laws and patterns govern the evolution of graphs. For example, an empirical study of social networks showed that they largely follow the law of triadic closure which suggests connections of nodes with mutual connections [20]. Similarly, the phenomenon of "rich get richer" signifying increasing connections to already popular nodes has been shown to hold true in social and information networks [21].

Understanding such underlying patterns is crucial as they become building blocks for developing algorithms for link prediction. Triadic closure property led to the development of "Common Neighbors" and "rich get richer" led to the development of "Preferential Attachment." Several other link prediction methods such as "Adamic-Adar" and "Jaccard Coefficient" are variants of the above methods aiming to capture the properties.

Using static graph embedding for prediction tasks can capture spatial patterns in the graphs as well as some of the above properties. Nodes with common neighbors can be embedded closer than ones without. This can help use triadic closure property. Similarly, high-degree nodes can be embedded separately from low-degree nodes supporting "rich get richer" phenomenon. However, graph embeddings can provide a much better platform to capture more complex patterns. As nodes are now embedded in a common space, we can use time series of embeddings from previous k time steps to understand the network more deeply.

For example, consider Fig. 4. On the left, we have a user who is browsing the web. On the right, we have a list of web pages he is browsing. He navigates from one web page to another by following the hyperlink on the web page he is on. If we use the above properties, we cannot predict the deletion of the first link and addition of second link. However, looking at multiple time steps, we can infer the pattern.

Capturing temporal patterns in graphs using graph embedding was pioneered by Goyal et al. [22]. The method uses nonlinear recurrent layers to capture temporal pattern. Furthermore, it uses an autoencoder to reduce the graph dimensionality before feeding into the recurrent layers. They propose three variants of the model by varying the input to recurrent layers and show the differences. Recently, several methods have extended this work. Some of them use Graph Convolutional

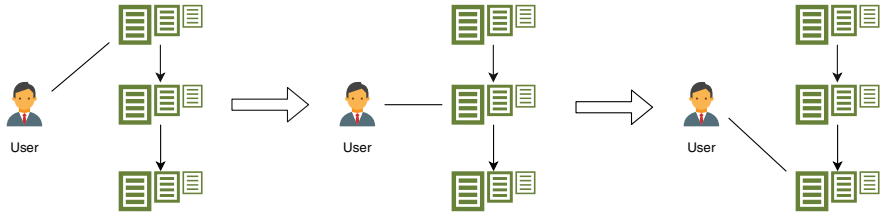


Fig. 4 Example illustrating a user browsing the web. The user follows the hyperlink on the web page to go to the next link

Network [23], while others use self-attention networks [25, 24] and other novelties in time series methods [26].

5 Attributed Graph Embedding Methods

Graphs are used as abstractions of real-world data. The richer the abstraction, the more properties of the underlying data it can capture. Representing a friendship graph as nodes and edges helps ease the analysis but loses important information such as user gender, location, and preferences which can govern formation of friendships. Similarly, in web network, capturing web page statistics and user demography is key in recommending new web pages.

Such information can be incorporated in the graph by storing it as node and edge attributes of the graph. For example, in social network, gender and location can be stored as user attribute and the type of relationship between the users can be incorporated as edge attributes. Extending graph embedding methods for such attributed graphs can ensure that the representation carries the attribute information.

Recently, many works have begun to use node attributes along with the network structure in the embedding process. Heterogeneous Network Embedding (HNE) [27] embeds the heterogeneous graph into a common space by learning separate embeddings of each category of node and then learning a matrix transformation to embed them into a common space. Another pioneer work in attributed graph embedding is Label Informed Attributed Network Embedding [28], which is a supervised method for learning the node representation given node labels and attributes. ELAINE [29] extended these methods to capture both node and edge attributes. It uses a coupled autoencoder to reconstruct both network features and node and edge attributes.

6 Conclusion

Graph embedding is a young and exciting field. This chapter presented some aspects of the timeline of this field. We started with graph embeddings of static graphs, understanding which properties are important to be captured and how to make the process scalable. We then extended this to dynamic graphs covering methods which update embeddings with new nodes and edge as well as methods which are capable of capturing dynamism in the graph. We then talked about attributed graph embedding methods which can embed more information in addition to nodes and edges.

Overall, the field is evolving continuously as the applications are enormous. The growth of this field is directly inspired by the use cases. Notwithstanding the recent innovations, there are several directions the field can grow. The first and foremost

is the interpretability of graph embeddings. Most methods produce embeddings which can be evaluated on the downstream tasks but are not interpretable. However, creating an interpretable embedding can yield tremendous insights into the dataset properties. Second, the methods have several hyperparameters that are difficult to tune. Coming up with automatic hyperparameter tuning methods can be another useful direction. Third, the field of dynamic graph embedding is still nascent. Temporal methods with good accuracy are often not scalable, and existing scalable methods do not capture long term graph dependencies. Bridging this gap is crucial to building a practical model.

References

1. Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering, 29(12), 2724–2743.
2. Goyal, P., and Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems, 151, 78–94.
3. Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., and Smola, A. J. (2013, May). Distributed large-scale natural graph factorization. In Proceedings of the 22nd international conference on World Wide Web (pp. 37–48). ACM.
4. Belkin, M., and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In Advances in neural information processing systems (pp. 585–591).
5. Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. (2016, August). Asymmetric transitivity preserving graph embedding. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1105–1114). ACM.
6. Cao, S., Lu, W., and Xu, Q. (2016, February). Deep neural networks for learning graph representations. In Thirtieth AAAI Conference on Artificial Intelligence.
7. Grover, A., and Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 855–864). ACM.
8. Wang, D., Cui, P., and Zhu, W. (2016, August). Structural deep network embedding. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1225–1234). ACM.
9. Kipf, T. N., and Welling, M. (2016). Variational graph auto-encoders. arXiv preprint arXiv:1611.07308.
10. Chamberlain, B. P., Clough, J., and Deisenroth, M. P. (2017). Neural embeddings of graphs in hyperbolic space. arXiv preprint arXiv:1705.10359.
11. Chami, I., Ying, Z., Ré, C., and Leskovec, J. (2019). Hyperbolic graph convolutional neural networks. In Advances in Neural Information Processing Systems (pp. 4869–4880).
12. Shen, X., Pan, S., Liu, W., Ong, Y. S., and Sun, Q. S. (2018, July). Discrete network embedding. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (pp. 3549–3555). AAAI Press.
13. Zhu, L., Guo, D., Yin, J., Ver Steeg, G., and Galstyan, A. (2016). Scalable temporal latent space inference for link prediction in dynamic social networks. IEEE Transactions on Knowledge and Data Engineering, 28(10), 2765–2777.
14. Zhang, Z., Cui, P., Pei, J., Wang, X., and Zhu, W. (2018, April). Timers: Error-bounded SVD restart on dynamic networks. In Thirty-Second AAAI Conference on Artificial Intelligence.
15. Goyal, P., Kamra, N., He, X., and Liu, Y. (2018). DynGEM: Deep embedding method for dynamic graphs. arXiv preprint arXiv:1805.11273.

16. Zhou, L., Yang, Y., Ren, X., Wu, F., and Zhuang, Y. (2018, April). Dynamic network embedding by modeling triadic closure process. In Thirty-Second AAAI Conference on Artificial Intelligence.
17. Rahman, M., Saha, T. K., Hasan, M. A., Xu, K. S., and Reddy, C. K. (2018). Dylink2vec: Effective feature representation for link prediction in dynamic networks. arXiv preprint arXiv:1804.05755.
18. Du, L., Wang, Y., Song, G., Lu, Z., and Wang, J. (2018, July). Dynamic Network Embedding: An Extended Approach for Skip-gram based Network Embedding. In IJCAI (pp. 2086–2092).
19. Ma, Y., Guo, Z., Ren, Z., Zhao, E., Tang, J., and Yin, D. (2018). Streaming Graph Neural Networks. arXiv preprint arXiv:1810.10627.
20. Kossinets, G., and Watts, D. J. (2006). Empirical analysis of an evolving social network. science, 311(5757), 88–90.
21. Easley, D., and Kleinberg, J. (2010). Networks, crowds, and markets (Vol. 8). Cambridge: Cambridge university press.
22. Goyal, P., Chhetri, S. R., and Canedo, A. (2018). dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. arXiv preprint arXiv:1809.02657.
23. Hajiramezanali, E., Hasanzadeh, A., Narayanan, K., Duffield, N., Zhou, M., and Qian, X. (2019). Variational graph recurrent neural networks. In Advances in Neural Information Processing Systems (pp. 10700–10710).
24. Sankar, A., Wu, Y., Gou, L., Zhang, W., and Yang, H. (2018). Dynamic Graph Representation Learning via Self-Attention Networks. arXiv preprint arXiv:1812.09430.
25. Sankar, A., Wu, Y., Gou, L., Zhang, W., and Yang, H. DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks.
26. Xu, D., Cheng, W., Luo, D., Liu, X., and Zhang, X. (2019, August). Spatio-temporal attentive RNN for node classification in temporal attributed graphs. In Proceedings of the 28th International Joint Conference on Artificial Intelligence (pp. 3947–3953). AAAI Press.
27. Chang, S., Han, W., Tang, J., Qi, G., Aggarwal, C., Huang, T. Heterogeneous network embedding via deep architectures.
28. Chang, S., Han, W., Tang, J., Qi, G. J., Aggarwal, C. C., and Huang, T. S. (2015, August). Heterogeneous network embedding via deep architectures. In Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 119–128). ACM.
29. Goyal, P., Hosseinmardi, H., Ferrara, E., and Galstyan, A. (2018). Capturing edge attributes via network embedding. IEEE Transactions on Computational Social Systems, 5(4), 907–917.
30. Katz, L. (1953). A new status index derived from sociometric analysis. Psychometrika, 18(1), 39–43.
31. Adamic, L. A., and Adar, E. (2003). Friends and neighbors on the web. Social networks, 25(3), 211–230.
32. Newman, M. E. (2001). Clustering and preferential attachment in growing networks. Physical review E, 64(2), 025102.
33. Lichtenwalter, R. N., Lussier, J. T., and Chawla, N. V. (2010, July). New perspectives and methods in link prediction. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 243–252).
34. Zhang, S., Tong, H., Xu, J., & Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. Computational Social Networks, 6(1), 11.

Autoencoders

Dor Bank, Noam Koenigstein, and Raja Giryes

1 Autoencoders

Autoencoders have been first introduced in [43] as a neural network that is trained to reconstruct its input. Their main purpose is learning in an unsupervised manner an "informative" representation of the data that can be used for various implications such as clustering. The problem, as formally defined in [2], is to learn the functions $A : \mathbb{R}^n \to \mathbb{R}^p$ (encoder) and $B : \mathbb{R}^p \to \mathbb{R}^n$ (decoder) that satisfy

$$\arg\min_{A,B} E[\Delta(\mathbf{x}, B \circ A(\mathbf{x})], \tag{1}$$

where E is the expectation over the distribution of x, and Δ is the reconstruction loss function, which measures the distance between the output of the decoder and the input. The latter is usually set to be the ℓ_2-norm. Figure 1 provides an illustration of the autoencoder model.

In the most popular form of autoencoders, A and B are neural networks [40]. In the special case that A and B are linear operations, we get a linear autoencoder [3]. In the case of linear autoencoder where we also drop the nonlinear operations, the autoencoder would achieve the same latent representation as Principal Component Analysis (PCA) [38]. Therefore, an autoencoder is in fact a generalization of PCA, where instead of finding a low-dimensional hyperplane in which the data lie, it is able to learn a nonlinear manifold.

D. Bank · R. Giryes
School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel
e-mail: dorbank@mail.tau.ac.il; raja@tauex.tau.ac.il

N. Koenigstein (✉)
Department of Industrial Engineering, Faculty of Engineering, Tel Aviv University, Tel Aviv, Israel
e-mail: noamk@tauex.tau.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_16

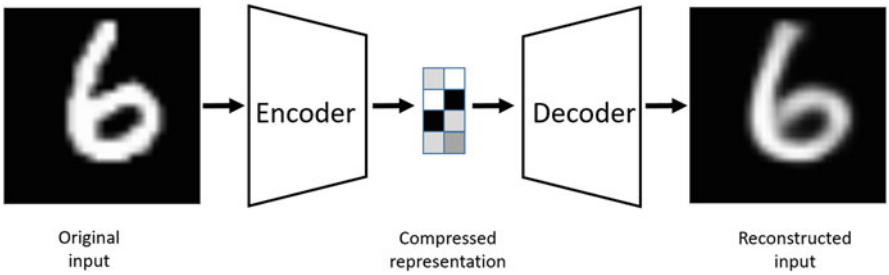


Fig. 1 An autoencoder example. The input image is encoded to a compressed representation and then decoded

Autoencoders may be trained end to end or gradually layer by layer. In the latter case, they are "stacked" together, which leads to a deeper encoder. In [35], this is done with convolutional autoencoders and in [54] with denoising autoencoder (described below).

This chapter is organized as follows. In Sect. 2, different regularization techniques for autoencoders are considered, whose goal is to ensure that the learned compressed representation is meaningful. In Sect. 3, the variational autoencoders are presented, which are considered to be the most popular form of autoencoders. Section 4 covers very common applications for autoencoders, Sect. 5.1 briefly discusses the comparison between autoencoders and generative adversarial networks, and Sect. 5 describes some recent advanced techniques in this field. Section 6 concludes this chapter.

2 Regularized Autoencoders

Since in training, one may just get the identity operator for A and B, which keeps the achieved representation the same as the input, some additional regularization is required. The most common option is to make the dimension of the representation smaller than the input. This way, a *bottleneck* is imposed. This option also directly serves the goal of getting a low-dimensional representation of the data. This representation can be used for purposes such as data compression, feature extraction, etc. It is important to note that even if the *bottleneck* is comprised of only one node, then overfitting is still possible if the capacity of the encoder and the decoder is large enough to encode each sample to an index.

In cases where the size of the hidden layer is equal to or greater than the size of the input, there is a risk that the encoder will simply learn the identity function. To prevent it without creating a bottleneck (i.e., smaller hidden layer), several options exist for regularization, which we describe hereafter, that would enforce the autoencoder to learn a different representation of the input.

An important trade-off in autoencoders is the bias-variance trade-off. On the one hand, we want the architecture of the autoencoder to be able to reconstruct the input well (i.e., reduce the reconstruction error). On the other hand, we want the low representation to generalize to a meaningful one. We now turn to describe different methods to tackle such trade-offs.

2.1 Sparse Autoencoders

One way to deal with this trade-off is to enforce sparsity on the hidden activations. This can be added on top of the bottleneck enforcement or instead of it. There are two strategies to enforce the sparsity regularization. They are similar to ordinary regularization, where they are applied on the activations instead of the weights. The first way to do so is to apply L_1 regularization, which is known to induce sparsity. Thus, the autoencoder optimization objective becomes

$$\arg\min_{A,B} E[\Delta(\mathbf{x}, B \circ A(\mathbf{x})] + \lambda \sum_i |a_i|, \tag{2}$$

where a_i is the activation at the ith hidden layer and i iterates over all the hidden activations. Another way to do so is to use the KL-divergence, which is a measure of the distance between two probability distributions. Instead of tweaking the $lambda$ parameter as in the L_1 regularization, we can assume the activation of each neuron acts as a Bernoulli variable with probability p and tweak that probability. At each batch, the actual probability is then measured, and the difference is calculated and applied as a regularization factor. For each neuron j, the calculated empirical probability is $\hat{p}_j = \frac{1}{m}\sum_i a_i(x)$, where i iterates over the samples in the batch. Thus, the overall loss function would be

$$\arg\min_{A,B} E[\Delta(\mathbf{x}, B \circ A(\mathbf{x})] + \sum_j KL(p||\hat{p}_j), \tag{3}$$

where the regularization term in it aims at matching p to $\hat{p}$.

2.2 Denoising Autoencoders

Denoising autoencoders [53] can be viewed either as a regularization option or as robust autoencoders which can be used for error correction. In these architectures, the input is disrupted by some noise (e.g., additive white Gaussian noise or erasures using dropout), and the autoencoder is expected to reconstruct the clean version of the input, as illustrated in Fig. 2.

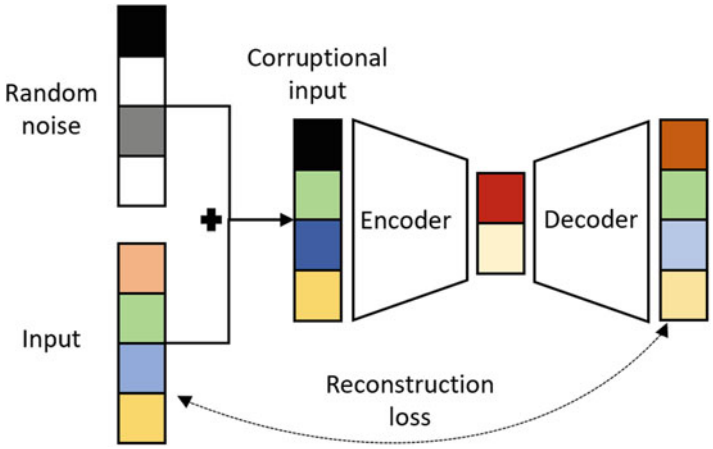


Fig. 2 A denoising autoencoder example. The disrupted input image is encoded to a representation and then decoded

Note that $\tilde{\mathbf{x}}$ is a random variable, whose distribution is given by $C(\tilde{\mathbf{x}}|\mathbf{x})$. Two common options for C are

$$C_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma^2 \mathcal{I}), \tag{4}$$

and

$$C_p(\tilde{\mathbf{x}}|\mathbf{x}) = \beta \odot \mathbf{x}, \quad \beta \sim Ber(p), \tag{5}$$

where $\odot$ denotes an element-wise (Hadamard) product. In the first option, the variance parameter σ sets the impact of the noise. In the second, the parameter p sets the probability of a value in $\mathbf{x}$ not being nullified. A relationship between denoising autoencoders with dropout to analog coding with erasures has been shown in [4].

2.3 Contractive Autoencoders

In denoising autoencoders, the emphasis is on letting the encoder be resistant to some perturbations of the input. In contractive autoencoders, the emphasis is on making the feature extraction less sensitive to small perturbations, by forcing the encoder to disregard changes in the input that are not important for the reconstruction by the decoder. Thus, a penalty is imposed on the Jacobian of the network. The Jacobian matrix of the hidden layer h consists of the derivative of each node h_j with respect to each value x_i in the input x. Formally, $J_{ji} = \nabla_{x_i} h_j(x_i)$. In contractive autoencoders, we try to minimize its L2 norm, such that the overall

optimization loss would be

$$\arg\min_{A,B} E[\Delta(x, B \circ A(x)] + \lambda ||J_A(x)||_2^2. \tag{6}$$

The reconstruction loss function and the regularization loss actually pull the result toward opposite directions. By minimizing the squared Jacobian norm, all the latent representations of the input tend to be more similar to each other and by thus make the reconstruction more difficult, since the differences between the representations are smaller. The main idea is that variations in the latent representation that are not important for the reconstructions would be diminished by the regularization factor, while important variations would remain because of their impact on the reconstruction error.

3 Variational Autoencoders

A major improvement in the representation capabilities of autoencoders has been achieved by the Variational Autoencoders (VAEs) model [27]. Following Variational Bayes (VB) Inference [6], VAEs are generative models that attempt to describe data generation through a probabilistic distribution. Specifically, given an observed dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ of V i.i.d. samples, we assume a generative model for each datum $\mathbf{x}_i$ conditioned on an unobserved random latent variable $\mathbf{z}_i$, where $\boldsymbol{\theta}$ are the parameters governing the generative distribution. This generative model is also equivalent to a *probabilistic decoder*. Symmetrically, we assume an approximate posterior distribution over the latent variable $\mathbf{z}_i$ given a datum $\mathbf{x}_i$ denoted by recognition, which is equivalent to a *probabilistic encoder* and governed by the parameters $\boldsymbol{\phi}$. Finally, we assume a prior distribution for the latent variables $\mathbf{z}_i$ denoted by $p_{\boldsymbol{\theta}}(\mathbf{z}_i)$. Figure 3 depicts the relationship described above. The parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are unknown and need to learn from the data. The observed latent variables $\mathbf{z}_i$ can be interpreted as a *code* given by the *recognition model* $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$.

The marginal log-likelihood is expressed as a sum over the individual data points $\log p_{\boldsymbol{\theta}}(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N) = \sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\mathbf{x}_i)$, and each point can be rewritten as

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) = D_{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}_i)\right) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i), \tag{7}$$

where the first term is the Kullback–Leibler divergence of the approximate recognition model from the true posterior and the second term is called the *variational lower bound* on the marginal likelihood defined as

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i) \triangleq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)}\Big[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\Big]. \tag{8}$$

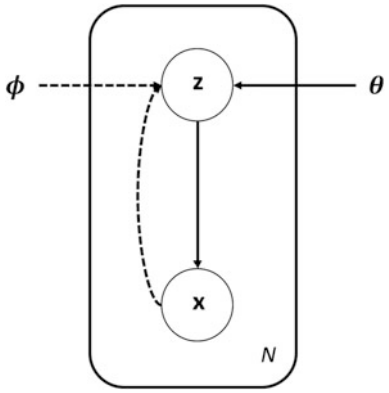


Fig. 3 A Graphical Representation of VAE

Since the Kullback–Leibler divergence is non-negative, $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i)$ is a lower bound on the marginal log-likelihood and the marginal log-likelihood is independent of the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, maximizing the lower bound improves our approximation of the posterior with respect to the Kullback–Leibler divergence.

The variational lower bound can be further expanded as follows:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i) = -D_{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)||p_{\boldsymbol{\theta}}(\mathbf{z})\right) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)}\Big[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z})\Big]. \tag{9}$$

Variational inference follows by maximizing $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i)$ for all data points with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

Given a dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ with N data points, we can estimate the marginal likelihood lower bound of the full dataset $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X})$ using a mini-batch $\mathbf{X}^M = \{\mathbf{x}_i\}_{i=1}^M$ of size M as follows:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) \approx \tilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M) = \frac{N}{M}\sum_{i=1}^{M}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i). \tag{10}$$

Classical mean-field VB assumes a factorized approximate posterior followed by a closed form optimization updates (which usually required conjugate priors). However, VAE follows a different path in which the gradients of $\tilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M)$ are approximated using the reparameterization trick and stochastic gradient optimization.

3.1 The Reparameterization Trick

The reparameterization trick is a simple approach to estimate $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i)$ based on a small sample of size L. Consider Eq. 8, and we can reparameterize the

random variable $\tilde{\mathbf{z}} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ using a differentiable transformation $g_{\boldsymbol{\phi}}(\epsilon, \mathbf{x})$ using an auxiliary noise variable ϵ drawn from some distribution $\epsilon \sim p(\epsilon)$ [27]. Using this technique, $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i)$ is approximated as follows:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i) \approx \tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i) = \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{z}_{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_{(i,l)|\mathbf{x}_i}), \tag{11}$$

where $\mathbf{z}_{(i,l)} = g_{\boldsymbol{\phi}}(\epsilon_{(i,l)}, \mathbf{x}_i)$ and $\epsilon_{(i,l)}$ is a random noise drawn from $\epsilon_l \sim p(\epsilon)$.

Remember we wish to optimize the mini-batch estimates from Eq. 10. By plugging Eq. 11, we get the following differentiable expression:

$$\hat{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) = \frac{N}{M} \sum_{i=1}^{M} \tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i), \tag{12}$$

which can be derived according to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ and plugged into an optimizer framework.

Algorithm 1 Pseudo-code for VAE

$(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow$ Initialize Parameter
repeat
  $\mathbf{X}^M \leftarrow$ Random mini-batch of M datapoints
  $\epsilon \leftarrow L$ random samples of $p(\epsilon)$
  $\mathbf{g} \leftarrow \nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \hat{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X})$ Gradients of Equation 12
  $(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow$ Update parameters based on $\mathbf{g}$ e.g., update with SGD or Adagrad
until Convergence of $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $(\boldsymbol{\theta}, \boldsymbol{\phi})$

Algorithm 1 summarizes the full optimization procedure for VAE. Often L can be set to 1 so long as M is large enough. Typical numbers are $M = 100$ and $L = 1$.

Equation 11 presents a lower bound on the log-likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{x}_i)$. In [8], the equation is changed to

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i) = \frac{1}{L} \sum_{l=1}^{L} \log \frac{1}{k} \sum_{j=1}^{k} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{z}_{(j,l)})}{q_{\boldsymbol{\phi}}(\mathbf{z}_{(j,l)|\mathbf{x}_i})}. \tag{13}$$

Intuitively, instead of taking the gradient of a single randomized latent representation, the gradients of the generative network are learned by a weighted average of the sample over different samples from its (approximated) posterior distribution. The weights simply the likelihood functions $q_{\boldsymbol{\phi}}(\mathbf{z}_{(j,l)|\mathbf{x}_i})$.

3.2 Example: The Case of Normal Distribution

Usually, we approximate $p(\mathbf{z}|\mathbf{x})$ with a Gaussian distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(g(\mathbf{x}), h(\mathbf{x}))$, where $g(\mathbf{x})$ and $h(\mathbf{x})$ are the mean and the covariance of the distribution defined by the encoder network. Namely, the encoder takes an input $\mathbf{x}_i$ and maps it into a mean and covariance that determine the approximate posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$.

To enable backpropagation through the network, sampling from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ can be simplified using the reparameterization trick as follows:

$$\mathbf{z} = h(\mathbf{x})\xi + g(\mathbf{x}), \tag{14}$$

where $\xi \sim \mathcal{N}(0, \mathbf{I})$ is a normal distribution.

Finally, we denote the decoder with an additional function f and require that $x \approx f(\mathbf{z})$. The loss function of the entire network then becomes

$$loss = c\,\|x - f(\mathbf{z})\|^2 + D_{KL}\left(\mathcal{N}(g(\mathbf{x}), h(\mathbf{x})), \mathcal{N}(0, \mathbf{I})\right), \tag{15}$$

which can be automatically derived with respect to the network parameters in g, h, and f and optimized with backpropagation.

3.3 Disentangled Autoencoders

The variational lower bound, as presented in Eq. 9, can be viewed as the summation of two terms: the right term that includes the reconstruction capability of samples, and the left term that acts as a regularization that biases $q_\phi(z|\mathbf{x}^{(i)}$ toward the assumed prior $p_\theta(z)$. Disentangled autoencoders are based on variational autoencoders with a small addition. They add a parameter β as a multiplicative factor for the KL-divergence [23] in Eq. 9. Its maximization factor is thus

$$\mathcal{L}(\theta, \phi, \mathbf{x}^{(i)}) = -\beta D_{KL}(q_\phi(z|\mathbf{x}^{(i)})||p_\theta(z)) + \mathbb{E}_{q_\phi(z|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}|z)]. \tag{16}$$

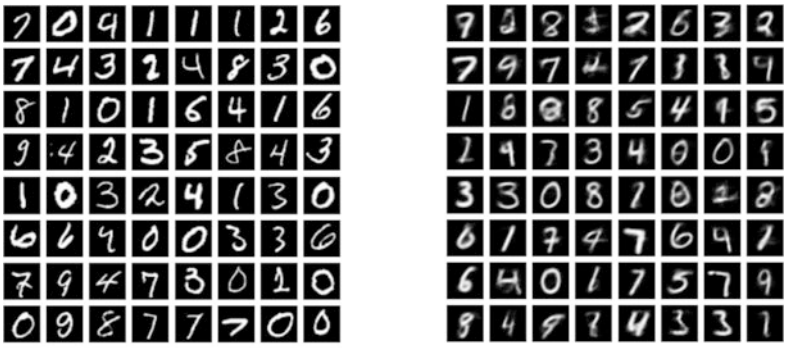
In practice, the prior $p_\theta(z)$ is commonly set as the standard multivariate normal distribution $\mathcal{N}(0, \mathcal{I})$. In those cases, all the features are uncorrelated, and the KL-divergence regularizes the latent features distribution $q_\phi(z|\mathbf{x}^{(i)}$ to a less correlated one. Note that the larger the β, the less correlated (more disentangled) the features will be.

4 Applications of Autoencoders

Learning a representation via the autoencoder can be used for various applications. The different types of autoencoders may be modified or combined to form new models for various applications. For example, in [39], they are used for classification, captioning, and unsupervised learning. We describe below some of the applications of autoencoders.

4.1 Autoencoders as a Generative Model

As explained in Sect. 3, variational autoencoders are generative models that attempt to describe data generation through a probabilistic distribution. Furthermore, as can be seen in Eq. 9, the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x}^{(i)}$ which is derived by the encoder is regularized toward a continuous and complete distribution in the shape of the predefined prior of the latent variables $p_{\boldsymbol{\theta}}(\mathbf{z})$. Once trained, one can simply sample random variables from the same prior and feed it to the decoder. Since the decoder was trained generate $\mathbf{x}$ from $p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z})$, it would generate a meaningful newly generated sample. In Fig. 4, original and generated images are displayed over the MNIST dataset. When discussing the generation of new samples, the immediate debate involves the comparison between VAE and GANs. An overview on this can be found in Sect. 5.1, and two methods that combine both models can be found in Sects. 5.2 and 5.3.



(a) (b)

Fig. 4 Generated images of a variational autoencoder, trained on the MNIST dataset with a prior $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(0, \mathcal{I})$. Left: original images from the dataset. Right: generated images. (**a**) Sample from the original MNIST dataset. (**b**) VAE generated MNIST images

4.2 Use of Autoencoders for Classification

While autoencoders are being trained in an unsupervised manner (i.e., in the absence of labels), they can be used also in the semi-supervised setting (where part of the data does have labels) for improving classification results. In this case, the encoder is used as a feature extractor and is "plugged" into a classification network. This is mainly done in the semi-supervised learning setup, where a large dataset is given for a supervised learning task, but only a small portion of it is labeled.

The key assumption is that samples with the same label should correspond to some latent presentation, which can be approximated by the latent layer of autoencoders. First, the autoencoders are trained in an unsupervised way, as described in previous sections. Then (or in parallel), the decoder is put aside, and the encoder is used as the first part of a classification model. Its weights may be fine-tuned [13] or stay fixed during training. A simpler strategy can be found in [17], where a support vector machine (SVM) is trained on the output features of the encoder. In cases where the domain is high dimensional, and the layer-by-layer training is unfeasible, one solution is to train each layer as a linear layer before adding the nonlinearity. In this case, even with denoising the inputs, there exists a closed form solution for each layer, and no iterative process is needed [9].

Another approach uses autoencoders as a regularization technique for a classification network. For example, in [29, 60], two networks are connected to the encoder, a classification network (trained with the labeled data) and the decoder network (trained to reconstruct the data, whether labeled or unlabeled). Having the reconstruction head in addition to the classification head serves a regularizer for the latter. An illustration is given in Fig. 5.

4.3 Use of Autoencoders for Clustering

Clustering is an unsupervised problem, where the target is to split the data to groups such that samples in each group are similar to one another and different from the samples in the other groups. Most of the clustering algorithms are sensitive to the dimensions of the data and suffer from the curse of dimensionality.

Assuming that the data have some low-dimensional latent representation, one may use autoencoders to calculate such representations for the data, which are composed of much less features. First, the autoencoder is trained as described in the sections before. Then, the decoder is put aside, similarly to the usage in classification. The latent representation (the encoders' output) of each data point is then kept and serves as the input for any given clustering algorithm (e.g., K-means).

The main disadvantage of using vanilla autoencoders for clustering is that the embeddings are trained solely for reconstruction and not for the clustering application. To overcome this, several modifications can be made. In [45], the clustering is done similarly to the K-means algorithm [55], but the embeddings

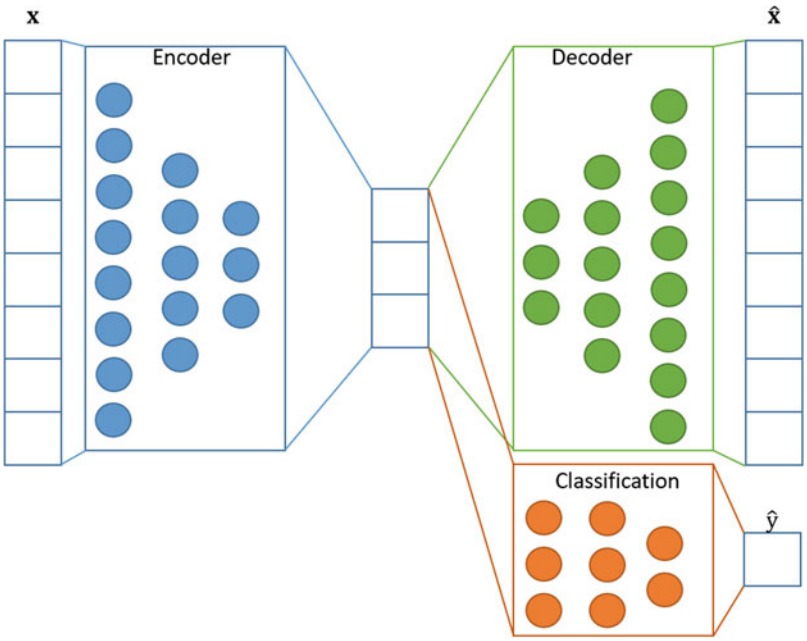


Fig. 5 An illustration for using autoencoders as regularization for supervised models. Given the reconstruction loss $R(x, \hat{x})$ and the classification lost function $\mathcal{L}(y, \hat{y})$, the new loss function would be $\tilde{\mathcal{L}} = \mathcal{L}(y, \hat{y}) + \lambda R(x, \hat{x})$, where λ is the regularization parameter

are also retrained at each iteration. In this training, an argument is added to the autoencoder loss function, which penalizes the distance between the embedding and the cluster center.

In [20], a prior distribution is made on the embeddings. Then, the optimization is done both by the reconstruction error and by the KL-divergence between the resulting embeddings distribution and the assumed prior. This can be done implicitly, by training a VAE with the assumed prior. In [10], this is done while assuming a multivariate Gaussian mixture.

4.4 Use of Autoencoders for Anomaly Detection

Anomaly detection is another unsupervised task, where the objective is to learn a normal profile given only the normal data examples and then identify the samples not conforming to the normal profile as anomalies. This can be applied in different applications such as fraud detection, system monitoring, etc. The use of autoencoders for this tasks follows the assumption that a trained autoencoder would

learn the latent subspace of normal samples. Once trained, it would result with a low reconstruction error for normal samples and high reconstruction error for anomalies [21, 18, 62, 61].

4.5 Use of Autoencoders for Recommendation Systems

A recommender system is a model or system that seeks to predict user preferences or affinities to items [41]. Recommender systems are prominent in e-commerce websites, application stores, and online content providers and have many other commercial applications. A classical approach in recommender system models is Collaborative Filtering (CF) [22]. In CF, user preferences are inferred based on information from other user preferences. The hidden assumption is that the human preferences are highly correlated, i.e., people who exhibit similar preferences in the past will exhibit similar preferences in the future.

A basic example of the use of autoencoders for recommender systems is the AutoRec model [44]. The AutoRec model has two variants: user-based AutoRec (U-AutoRec) and item-based AutoRec (I-AutoRec). In U-AutoRec, the autoencoder learns a lower dimensional representation of item preferences for specific users, while in I-AutoRec the autoencoder learns a lower dimensional representation of user preferences for specific items.

For example, assume a dataset consisting of M user and N items. Let $\mathbf{r}_m \in \mathcal{R}^N$ be a preference vector for the user m consisting of its preference score to each of the N items. U-AutoReco's decoder is $z = g(\mathbf{r}_m)$ mapping $\mathbf{r}_m$ into representation the representation vector $z \in \mathcal{R}^d$, where $d \ll N$. The reconstruction given the encoder $f(z)$ is $h(\mathbf{r}_m; \theta) = f(g(\mathbf{r}_m))$, where θ are the model's parameters. The U-AutoRec objective is defined as

$$\arg\min_\theta \sum_{m=1}^{M} \|\mathbf{r}_m - h(\mathbf{r}_m; \theta)\|_O^2 + \lambda \cdot reg. \tag{17}$$

Here, $\|\cdot\|_O^2$ means that the loss is defined only on the observed preferences of the user. At prediction time, we can investigate the reconstruction vector and find items that the user is likely to prefer.

The I-AutoRec is defined symmetrically as follows: Let $\mathbf{r}_n$ be item n's preference vector for each user. The I-AutoRec objective is defined as

$$\arg\min_\theta \sum_{n=1}^{N} \|\mathbf{r}_n - h(\mathbf{r}_n; \theta)\|_O^2 + \lambda \cdot reg. \tag{18}$$

At prediction time, we reconstruct the preference vector for each item and look for potential users with high predicted preference.

In [47, 46], the basic AutoRec model was extended by including denoising techniques and incorporating users and items side information such as user demographics or item description. The denoising techniques serve as another type of regularization that prevents the autoencoder overfitting rare patterns that do not concur with general user preferences. The side information was shown to improve accuracy and speed up the training process.

Similar to the original AutoRec, two symmetrical models have been proposed: one that works with user preference $\mathbf{r}_m$ vectors and the other with item preference vectors $\mathbf{r}_n$. In the general case, these vectors may consist of explicit ratings. The Collaborative Denoising Autoencoder (CDAE) model [59] is essentially applying the same approach on vectors of implicit ratings rather than explicit ratings. Finally, a variational approach has been attempted by applying VAE in a similar fashion [31].

4.6 Use of Autoencoders for Dimensionality Reduction

Real-world data such as text or images are often represented using a sparse high-dimensional representation. While many models and applications work directly in the high-dimensional space, this often leads to the *curse of dimensionality* [15]. The goal of dimensionality reduction is to learn a lower dimensional manifold, so-called intrinsic dimensionality space.

A classical approach for dimensionality reduction is Principal Component Analysis (PCA) [58]. PCA is a linear projection of data points into a lower dimensional space such that the squared reconstruction loss is minimized. As a linear projection, PCA is optimal. However, nonlinear methods such as autoencoders may and often do achieve superior results.

Other methods for dimensionality reduction employ different objectives. For example, Linear Discriminant Analysis (LDA) is a supervised method to find a linear subspace, which is optimal for discriminating data from different classes [11]. ISOMAP [48] learns a low-dimensional manifold by retaining the geodesic distance between pairwise data in the original space. For a survey of different dimensionality methods, see [51].

The use of autoencoders for dimensionality reduction is straightforward. In fact, the dimensionality reduction is performed by every autoencoder in the bottleneck layer. The projection of the original input into the lower dimensional bottleneck representation is a dimension reduction operation through the encoder and under the objective given to the decoder. For example, an autoencoder comprised of a simple fully connected encoder and decoder with a squared loss objective performs dimension reduction with a similar objective to PCA. However, the nonlinearity activation functions often allow for a superior reconstruction when compared to simple PCA. More complex architectures and different objectives allow different complex dimension reduction models. To review the different applications of autoencoders for dimension reduction, we refer the interested reader to [24, 56, 57].

5 Advanced Autoencoder Techniques

Autoencoders are usually trained by a loss function corresponding to the difference between the input and the output. As shown above, one of the strengths of autoencoders is the ability to use their latent representation for different usages. On the other hand, by looking at the reconstruction quality of autoencoders for images, one of its major weaknesses becomes clear, as the resulting images are usually blurry. The reason for that is the used loss function, which does not take into account how realistic its results are and does not use the prior knowledge that the input images are not blurred. In recent years, there were some developments related to autoencoders, which deal with this weakness.

5.1 Autoencoders and Generative Adversarial Networks

Variational autoencoders are trained (usually) on MSE which yields slightly blurred images but allows inference over the latent variables in order to control the output. An alternative generative model to autoencoders that synthesize data (such as images) is the Generative Adversarial Networks (GANs). In a nutshell, a GAN architecture consists of two parts: the generator that generates new samples, and a discriminator that is trained to distinguish between real samples and generated ones. The generator and the discriminator are trained together using a loss function that enforces them to compete with each other and by thus improves the quality of the generated data. This leads to generated results that are quite compelling visually, but in the cost of the control on the resulting images. Different works have been done for having the advantages of both models, by different combinations of the architectures and the loss functions. In Adversarial Autoencoders [34], the KL-divergence in the VAE loss function is replaced by a discriminator network that distinguishes between the prior and the approximated posterior. In [28], the reconstruction loss in the VAE loss is replaced by a discriminator, which makes the decoder to essentially merge with the generator. In [14], the discriminator of the GAN is combined with an encoder via shared weights, which enables the latent space to be conveniently modeled by GMM for inference. This approach was then used in [25] for self-supervised learning. We detail next two other directions for combining GANs with autoencoders.

5.2 Adversarially Learned Inference

One of the disadvantages of GANs is mode collapse, which unlike autoencoders may cause them to represent via the latent space just part of the data (miss some modes in its distribution) and not all of it.

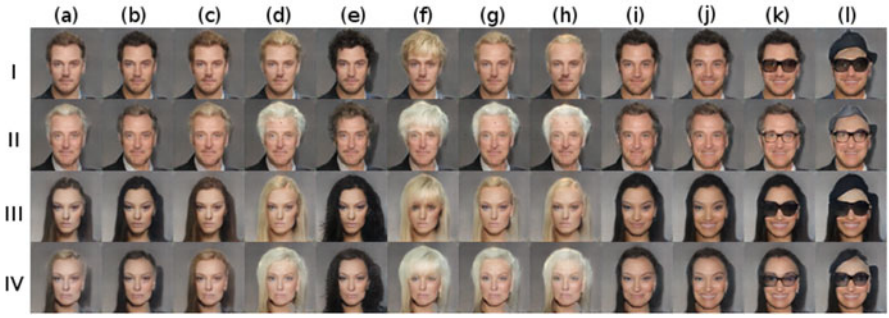


Fig. 6 An Image drawn from [12]. A model is first trained on the CelebA dataset [33]. It includes 40 different attributes on each image, which in ALI are linearly embedded in the encoder, decoder, and discriminator. Following the training phase, a single fixed latent code z is sampled. Each row has a subset of attributes that are held constant across columns. The attributes are male, attractive, young for row I; male attractive, older for row II; female, attractive, young for row III; and female, attractive, older for Row IV. Attributes are then varied uniformly over rows across all columns in the following sequence: (**b**) black hair; (**c**) brown hair; (**d**) blond hair; (**e**) black hair, wavy hair; (**f**) blond hair, bangs; (**g**) blond hair, receding hairline; (**h**) blond hair, balding; (**i**) black hair, smiling; (**j**) black hair, smiling, mouth slightly open; (**k**) black hair, smiling, mouth slightly open, eyeglasses; and (**l**) black hair, smiling, mouth slightly open, eyeglasses, wearing hat

In Adversarially Learned Inference (ALI), there is an attempt to merge the ideas of both VAEs and GANS and get a compromise of their strengths and weaknesses [12]. Instead of training a VAE with some loss function between the input and the output, a discriminator is used to distinguish between $(\mathbf{x}, \hat{\mathbf{z}})$ pairs, where $\mathbf{x}$ is an input sample and $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$ is sampled from the encoders output, and $(\tilde{\mathbf{x}}, \mathbf{z})$ pairs, where $\mathbf{z} \sim p(\mathbf{z})$ is sampled from the used prior in the VAE, and $\tilde{\mathbf{x}} \sim p(\mathbf{x}|\mathbf{z})$ is the decoder output. This way the decoder is enforced to output realistic results in order to "fool" the discriminator. Yet, the autoencoder structure is maintained. An example of how ALI enables altering specific features in order to get meaningful alterations in images is presented in Fig. 6.

ALI is an important milestone in the goal of merging both concepts and it had many extensions. For example, HALI [5] learns the autoencoder in hierarchical structure in order to improve the reconstruction ability. ALICE [30] added a conditional entropy loss between the real and the reconstructed images.

5.3 Wasserstein Autoencoders

In continuation to Sect. 5.2, GANs generate compelling images but do not provide inference and have a lot of inherent problems regarding its learning stability. Wasserstein GAN (WGAN) [1] solves a lot of those problems by using the Wasserstein distance for the optimizations loss function. The Wasserstein distance is a specific case of the Optimal Transport distance [52], which is a distance between

two probabilities, P_X and P_G, and is defined as

$$W_c(P_X, P_G) = \inf_{\Gamma \in P(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y)\sim\Gamma}[c(X, Y)], \tag{19}$$

where $c(x, y)$ is some cost function. When $c(x, y) = d^p(x, y)$ is a metric measurement, then the p-th root of W_c is called the p-Wasserstein distance. When $c(x, y) = d(x, y)$, then we get to the 1-Wasserstein distance, which is also known as the "Earth Moving Distance" [42] and can be defined as

$$W_1(P_X, P_G) = \sup_{f \in \mathfrak{F}} \mathbb{E}_{X\sim P_X}[f(X)] - \mathbb{E}_{Y\sim P_G}[f(Y)]. \tag{20}$$

Informally, we try to match the two probabilities by "moving" the first to the latter in the shortest distance, and that distance is defined as the 1-Wasserstein distance.

As seen in Eq. 9, the loss function of a specific sample is comprised of the reconstruction error and a regularization factor which enforces the latent representation to resemble the prior (usually multivariate standard normal). The problem addressed in [49] is that this regularization essentially pushes all the samples to look the same and does not use the entire latent space as a whole. In GANs, the OT distance is used to discriminate between the distribution of real images and the distribution of fake ones. In Wasserstein autoencoders (WAEs) [49], the authors modified the loss function for autoencoders, which leads to the following objective:

$$D_{WAE}(P_X, P_G) = \inf_{Q(Z|X)\in\mathfrak{Q}} \mathbb{E}_{P_X}\mathbb{E}_{Q(Z|X)}[c(X, G(Z))] + \lambda \cdot D_Z(Q_Z, P_Z), \tag{21}$$

where Q is the encoder and G is the decoder. The left part is the new reconstruction loss, which now penalizes on the output distribution and the sample distribution. This is penalization since the "transportation plan" factors through the G mapping [7]. The right part penalizes the distance between the latent space distribution and the prior distribution. The authors keep the prior as the multivariate normal distribution and use to examples for divergences: the Jensen–Shannon divergence $D_j s$ [32] and the maximum mean discrepancy (MMD) [19]. Figure 7 illustrates the regularizations difference between VAE and WAE.

5.4 Deep Feature Consistent Variational Autoencoder

In this section, a different loss function is presented to optimize the autoencoder. Given an original image and a reconstructed one, instead of measuring some norm on the pixel difference (such as the ℓ_2), a different measure is used that takes into account the correlation between the pixels.

Pretrained classification networks are commonly used for transfer learning. They allow transcending between different input domains, where the weights of the

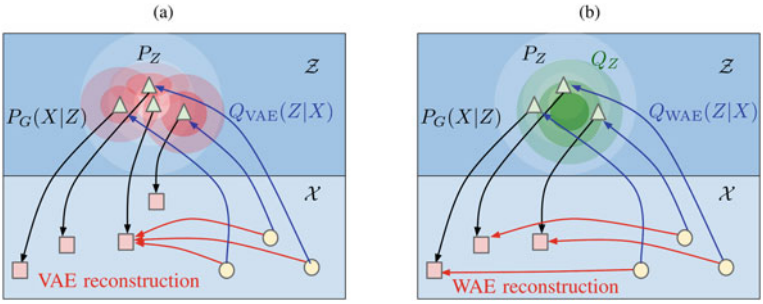


Fig. 7 An Image drawn from [49]. Both VAE and WAE minimize two terms: the reconstruction cost and the regularizer penalizing discrepancy between P_Z and distribution induced by the encoder Q. VAE forces $Q(Z|X = x)$ to match P_Z for all the different input examples x drawn from P_X. This is illustrated on picture (**a**), where every single red ball is forced to match P_Z depicted as the white shape. Red balls start intersecting, which leads to problems with reconstruction. In contrast, WAE forces the continuous mixture $Q_Z := \int Q(Z|X)dP_X$ to match P_Z, as depicted with the green ball in picture (**b**). As a result, latent codes of different examples get a chance to stay far away from each other, promoting a better reconstruction. (**a**) VAE. (**b**) WAE

model, which have been trained for one domain, are fine tuned for the new domain in order to adapt to the changes between the domains. This can be done by training all the models' (pretrained) weights for several epochs or just the final layers. Another use of pretrained networks is style transfer, where a style of one image is transferred to another image [16], e.g., causing a regular photo looks like a painting of a given painter (e.g., Van Gogh) while maintaining its content (e.g., keeping the trees, cars, houses, etc. at the same place). In this case, the pretrained networks serve as a loss function.

The same can be done for autoencoders. A pretrained network can be used for creating a loss function for autoencoders [26]. After encoding and decoding an image, both the original and reconstructed images are inserted as input to a pretrained network. Assuming the pretrained network results with high accuracy and the domain that it was trained on is not too different than the one of the autoencoder, then each layer can be seen as a successful feature extractor of the input image. Therefore, instead of measuring the difference between the two images directly, it can be measured between their representation in the network layers. The difference between the images at different layers in the network imposes a more realistic difference measure for the autoencoder.

5.5 Conditional Image Generation with PixelCNN Decoders

Another alternative proposes a composition between autoencoders and PixelCNN [37]. In PixelCNN [36], the pixels in the image are ordered by some arbitrary order (e.g., top to bottom, left to right, or RGB values). Then, the output is formed

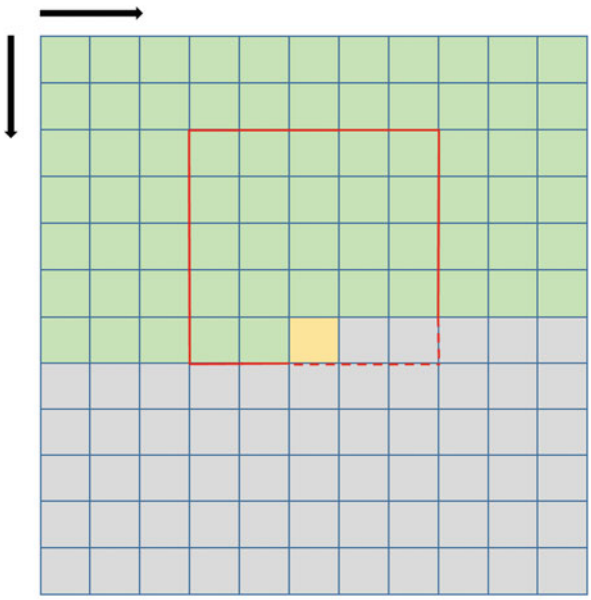


Fig. 8 The pixelCNN generation framework. The pixels are generated sequentially. In this case, they are generated from top to bottom and from left to right. The next pixel to be generated is the yellow one. The green pixels are the already generated ones. For generating the yellow pixel, the pixelRNN takes into account the hidden state and the information of the green pixels in the red square

sequentially where each pixel is a result of both the output of previous pixels and the input. This strategy takes into account the local spatial statistics of the image, as illustrated in Fig. 8. For example, below a background pixel, there is a higher chance to have another background pixel than the chance of having a foreground pixel. With the use of the spatial ordering (in addition to the input pixel information), the probability of getting a blurred pixel diminishes. In a later development [50], the local statistics was replaced by the usage of an RNN, but the same concept of pixel generation was remained. This concept can be combined with autoencoders by setting the decoder to be structured as a pixelCNN network generating the output image in a sequential order.

6 Conclusion

This chapter presented autoencoders showing how the naive architectures that were first defined for them evolved to powerful models with the core abilities to learn a meaningful representation of the input and to model generative processes. These two abilities can be easily transformed to various use-cases, where part of them was covered. As explained in Sect. 5.2, one of the autoencoders fallbacks is that its reconstruction errors do not include how realistic the outputs are. As for modeling generative processes, despite the success of variational and disentangled autoencoders, the way to choose the size and distribution of the hidden state is still based on experimentation, by considering the reconstruction error and by varying

the hidden state at post training. A future research that better sets these parameters is required.

To conclude, the goal of autoencoders is to get a compressed and meaningful representation. We would like to have a representation that is meaningful to us and at the same time good for reconstruction. In that trade-off, it is important to find the architecture which serves all needs.

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: D. Precup, Y.W. Teh (eds.) Proceedings of the 34th International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 70, pp. 214–223. PMLR, International Convention Centre, Sydney, Australia (2017)
2. Baldi, P.: Autoencoders, unsupervised learning, and deep architectures. In: I. Guyon, G. Dror, V. Lemaire, G. Taylor, D. Silver (eds.) Proceedings of ICML Workshop on Unsupervised and Transfer Learning, *Proceedings of Machine Learning Research*, vol. 27, pp. 37–49. PMLR, Bellevue, Washington, USA (2012)
3. Baldi, P., Hornik, K.: Neural networks and principal component analysis: Learning from examples without local minima. Neural Netw. **2**(1), 53–58 (1989). https://doi.org/10.1016/0893-6080(89)90014-2
4. Bank, D., Giryes, R.: An ETF view of dropout regularization. In: 31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7–10, 2020. BMVA Press (2020). https://www.bmvc2020-conference.com/assets/papers/0044.pdf
5. Belghazi, M.I., Rajeswar, S., Mastropietro, O., Rostamzadeh, N., Mitrovic, J., Courville, A.: Hierarchical adversarially learned inference (2018)
6. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg (2006)
7. Bousquet, O., Gelly, S., Tolstikhin, I., Simon-Gabriel, C.J., Schoelkopf, B.: From optimal transport to generative modeling: the vegan cookbook. arXiv (2017)
8. Burda, Y., Grosse, R.B., Salakhutdinov, R.: Importance weighted autoencoders. CoRR **abs/1509.00519** (2015)
9. Chen, M., Xu, Z., Weinberger, K., Sha, F.: Marginalized denoising autoencoders for domain adaptation. Proceedings of the 29th International Conference on Machine Learning, ICML 2012 **1** (2012)
10. Dilokthanakul, N., Mediano, P.A.M., Garnelo, M., Lee, M.C.H., Salimbeni, H., Arulkumaran, K., Shanahan, M.: Deep unsupervised clustering with gaussian mixture variational autoencoders. ArXiv **abs/1611.02648** (2017)
11. Duda, R.O., Hart, P.E., Stork, D.G., et al.: Pattern classification. International Journal of Computational Intelligence and Applications **1**, 335–339 (2001)
12. Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., Courville, A.C.: Adversarially learned inference. ArXiv **abs/1606.00704** (2016)
13. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. **11**, 625–660 (2010)
14. Feigin, Y., Spitzer, H., Giryes, R.: GMM-based generative adversarial encoder learning (2020)
15. Friedman, J.H.: On bias, variance, 0/1—loss, and the curse-of-dimensionality. Data mining and knowledge discovery **1**(1), 55–77 (1997)
16. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2414–2423 (2016)

17. Gogoi, M., Begum, S.A.: Image classification using deep autoencoders. In: 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pp. 1–5 (2017). https://doi.org/10.1109/ICCIC.2017.8524276
18. Gong, D., Liu, L., Le, V., Saha, B., Mansour, M.R., Venkatesh, S., van den Hengel, A.: Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection (2019)
19. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.: A kernel two-sample test. J. Mach. Learn. Res. **13**(null), 723–773 (2012)
20. Guo, X., Liu, X., Zhu, E., Yin, J.: Deep clustering with convolutional autoencoders. pp. 373–382 (2017)
21. Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A.K., Davis, L.S.: Learning temporal regularity in video sequences. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 733–742 (2016)
22. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining Collaborative Filtering Recommendations. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, CSCW '00, pp. 241–250. ACM (2000)
23. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M.M., Mohamed, S., Lerchner, A.: beta-VAE: Learning basic visual concepts with a constrained variational framework. In: ICLR (2017)
24. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. science **313**(5786), 504–507 (2006)
25. Hochberg, D.C., Giryes, R., Greenspan, H.: A self supervised Stylegan for Classification with extremely limited annotations (2021)
26. Hou, X., Shen, L., Sun, K., Qiu, G.: Deep feature consistent variational autoencoder. CoRR **abs/1610.00291** (2016). http://arxiv.org/abs/1610.00291
27. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. CoRR **abs/1312.6114** (2013)
28. Larsen, A.B.L., Sønderby, S.K., Larochelle, H., Winther, O.: Autoencoding beyond pixels using a learned similarity metric. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning—Volume 48, ICML'16, p. 1558–1566. JMLR.org (2016)
29. Le, L., Patterson, A., White, M.: Supervised autoencoders: Improving generalization performance with unsupervised regularizers. In: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (eds.) Advances in Neural Information Processing Systems 31, pp. 107–117. Curran Associates, Inc. (2018)
30. Li, C., Liu, H., Chen, C., Pu, Y., Chen, L., Henao, R., Carin, L.: Alice: Towards understanding adversarial learning for joint distribution matching. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017). https://proceedings.neurips.cc/paper/2017/file/ade55409d1224074754035a5a937d2e0-Paper.pdf
31. Liang, D., Krishnana, R.G., Hoffman, M.D., Jebara, T.: Variational autoencoders for collaborative filtering. CoRR **abs/1802.05814** (2018). https://arxiv.org/abs/1802.05814
32. Lin, J.: Divergence measures based on the Shannon entropy. IEEE Transactions on Information Theory **37**(1), 145–151 (1991). https://doi.org/10.1109/18.61115. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=61115
33. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV '15, p. 3730–3738. IEEE Computer Society, USA (2015). https://doi.org/10.1109/ICCV.2015.425
34. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I.J.: Adversarial autoencoders. CoRR **abs/1511.05644** (2015)
35. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: T. Honkela, W. Duch, M. Girolami, S. Kaski (eds.) Artificial Neural Networks and Machine Learning—ICANN 2011, pp. 52–59. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
36. van den Oord, A., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks (2016)

37. Oord, A.v.d., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., Kavukcuoglu, K.: Conditional image generation with pixelCNN decoders. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, pp. 4797–4805. Curran Associates Inc., USA (2016). http://dl.acm.org/citation.cfm?id=3157382.3157633
38. Plaut, E.: From principal subspaces to principal components with linear autoencoders (2018)
39. Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., Carin, L.: Variational autoencoder for deep learning of images, labels and captions. In: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pp. 2352–2360 (2016)
40. Ranzato, M., Huang, F.J., Boureau, Y., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007). https://doi.org/10.1109/CVPR.2007.383157
41. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender systems handbook, pp. 1–35. Springer (2011)
42. Rubner, Y., Tomasi, C., Guibas, L.: The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision **40**, 99–121 (2000). https://doi.org/10.1023/A:1026543900054
43. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chap. Learning Internal Representations by Error Propagation, pp. 318–362. MIT Press, Cambridge, MA, USA (1986). http://dl.acm.org/citation.cfm?id=104279.104293
44. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: AutoRec: Autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18–22, 2015—Companion Volume, pp. 111–112 (2015)
45. Song, C., Liu, F., Huang, Y., Wang, L., Tan, T.: Auto-encoder based data clustering. In: J. Ruiz-Shulcloper, G. Sanniti di Baja (eds.) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 117–124. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
46. Strub, F., Mary, J.: Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs. In: NIPS Workshop on Machine Learning for eCommerce. Montreal, Canada (2015)
47. Strub, F., Mary, J., Gaudel, R.: Hybrid recommender system based on autoencoders. CoRR **abs/1606.07659** (2016). http://arxiv.org/abs/1606.07659
48. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. science **290**(5500), 2319–2323 (2000)
49. Tolstikhin, I., Bousquet, O., Gelly, S., Scholkopf, B.: Wasserstein auto-encoders. ICML 2018 (2018)
50. Van Den Oord, A., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning—Volume 48, ICML'16, p. 1747–1756. JMLR.org (2016)
51. Van Der Maaten, L., Postma, E., Van den Herik, J.: Dimensionality reduction: a comparative review. J Mach Learn Res **10**, 66–71 (2009)
52. Villani, C.: Topics in Optimal Transportation. Graduate studies in mathematics. American Mathematical Society (2003). https://books.google.co.il/books?id=GqRXYFxe0l0C
53. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning, ICML '08, pp. 1096–1103. ACM, New York, NY, USA (2008). https://doi.org/10.1145/1390156.1390294. http://doi.acm.org/10.1145/1390156.1390294
54. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010). http://dl.acm.org/citation.cfm?id=1756006.1953039
55. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S.: Constrained k-means clustering with background knowledge. In: Proceedings of the Eighteenth International Conference on

Machine Learning, ICML '01, p. 577–584. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
56. Wang, W., Huang, Y., Wang, Y., Wang, L.: Generalized autoencoder: A neural network framework for dimensionality reduction. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 490–497 (2014)
57. Wang, Y., Yao, H., Zhao, S.: Auto-encoder based dimensionality reduction. Neurocomputing **184**, 232–242 (2016)
58. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. Chemometrics and intelligent laboratory systems **2**(1–3), 37–52 (1987)
59. Wu, Y., DuBois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22–25, 2016, pp. 153–162 (2018)
60. Zhang, Y., Lee, K., Lee, H.: Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning—Volume 48, ICML'16, p. 612–621. JMLR.org (2016)
61. Zhao, Y., Deng, B., Shen, C., Liu, Y., Lu, H., Hua, X.S.: Spatio-temporal autoencoder for video anomaly detection. In: Proceedings of the 25th ACM International Conference on Multimedia, MM '17, p. 1933–1941. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3123266.3123451
62. Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., ki Cho, D., Chen, H.: Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: ICLR (2018)

Generative Adversarial Networks

Gilad Cohen and Raja Giryes

1 Introduction to GANs

Generative adversarial networks (GANs) are currently the leading method to learn a distribution of a given dataset and generate new examples from it. To begin to understand the concept of GANs, let us consider an interplay between the police and money counterfeiters. A state just launched its new currency and millions of bills and coins are widespread all over the country. Recently, the police detected a flood of counterfeit money in circulation. Further inspection reveals that the forged bills are lighter than the genuine bills and can be easily filtered out. The criminals then find out about the police's discovery and use new printers which control better the cash weight. In turn, the police investigate and discover that the new counterfeit bills have a different texture near the corners and remove them from the system. After many iterations of generating and discriminating forged money, the counterfeit money becomes almost indistinguishable from the real money. Note that the system has two agents: (1) the counterfeiters who create "close to real" money and (2) the police who detect counterfeit bills adequately.

Back to deep learning, the above two agents are called "generator" and "discriminator," respectively. These two entities are trained jointly, where the generator learns how to fool the discriminator with new adversarial examples out of the dataset distribution, and the discriminator learns to distinguish between real and fake data samples. The GAN architecture is used in more and more applications since its introduction in 2014. It was proved successful in many domains such as computer vision [14, 29, 25, 38], semantic segmentation [39, 27, 70, 24], time-series synthesis [9, 23], image editing [61, 36, 19, 3, 75], natural language processing [15, 28, 22],

G. Cohen · R. Giryes (✉)
Tel Aviv University, Tel Aviv, Israel
e-mail: giladco1@post.tau.ac.il; raja@tauex.tau.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_17

text-to-image generation [59, 58, 54], and many more. In the next section, we depict the basic GAN architecture and loss. Later, we will present more sophisticated architectures, losses, and common usages.

2 The Basic GAN Concept

The first GAN that was introduced by Goodfellow et al. [20] is depicted in Fig. 1.

The architecture of GANs is comprised of two individual components: a discriminator (D) and a generator (G). D is trained to distinguish between real images from the natural distribution and generated images, while G is trained to craft fake images that fool the discriminator. A random distribution, $\mathbf{z} \sim p_{\mathbf{Z}}$, is given as input to G. The purpose of GANs is to learn the generated samples' distribution, $G(\mathbf{z}) \sim p_g$ that estimates the real world distribution p_r. GANs are optimized by solving the following min–max optimization problem:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_r} \log[D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{Z}}} \log[1 - D(G(\mathbf{z}))]. \tag{1}$$

On the one hand, D aims at predicting $D(\mathbf{x}) = 1$ for real data samples and $D(G(\mathbf{z})) = 0$ for fake samples. On the other hand, the GAN learns how to fool D by finding G which is optimized on hampering the second term in Eq. (1).

On the first iteration, only the discriminator weights θ_D are updated. We sample a minibatch of m noise samples $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(m)}\}$ from $p_{\mathbf{Z}}$ and a minibatch of m real data examples $\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(m)}\}$ from p_r. We then calculate the discriminator's gradients

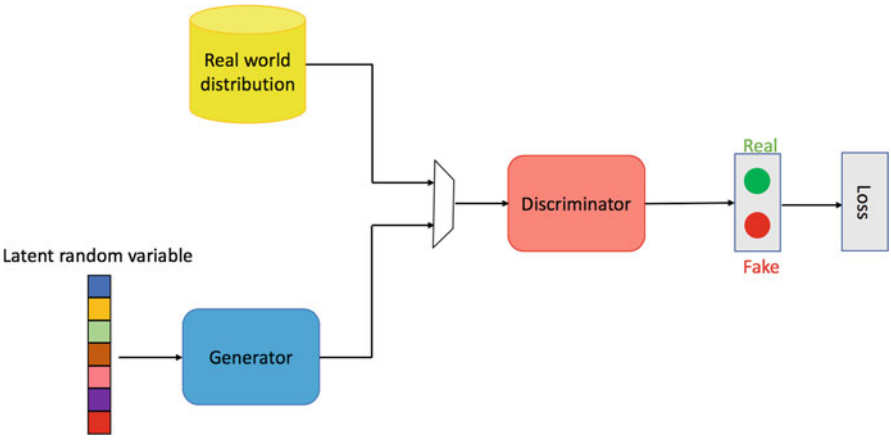


Fig. 1 Basic GAN structure. The discriminator and the generator are two deep neural networks trained jointly. The discriminator is trained for the task of classifying whether an input image is natural (real) or generated (fake), while the generator is optimized to fool the discriminator

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\mathbf{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\mathbf{z}^{(i)}\right)\right)\right) \right]$$

and update the discriminator weights, θ_D, by ascending this term. On the second iteration, only the generator's weights θ_G are updated. We sample a minibatch of m noise samples $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(m)}\}$ from $p_{\mathbf{z}}$, calculate the generator's gradients

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\mathbf{z}^{(i)}\right)\right)\right), \tag{2}$$

and update the generator weights, θ_G, by descending this term. Goodfellow et al. [20] showed that under certain conditions on D, G, and the training procedure, the distribution $p_{\mathbf{z}}$ converges to p_r.

3 GAN Advantages and Problems

Since their first introduction in 2014, GANs have attracted a growing interest all over the academia and industry, thanks to many advantages over other generative models (mainly Variational Auto-encoders (VAEs) [34]):

- **Sharp images**: GANs produce sharper images than other generative models. The images at the output of the generator look more natural and with better quality than images generated using VAEs, which tend to be blurrier.
- **Configurable size**: The latent random variable size is not restricted, enriching the generator search space.
- **Versatile generator**: The GAN framework can support many different generator networks, unlike other generative models that may have architectural constraints. VAEs, for example, enforce using a Gaussian at the generator's first layer.

The above advantages make GANs very attractive in the deep learning community, achieving state-of-the-art results in a variety of domains and generating very natural images. Yet, the original GAN suffers from three major problems that are described in detail below. We first present a short summary of them:

- **Mode collapse**: During the synchronized training of the generator and discriminator, the generator tends to learn to produce a specific pattern (mode) which fools the discriminator. Although this pattern minimizes Eq. (1), the generator does not cover the full distribution of the dataset.
- **Vanishing gradients**: Very frequently, the discriminator is trained "too well" to distinguish real images from adversarial images; in this scenario, the training step of the generator back propagates very low gradients, which does not help the generator to learn.

- **Instability**: The model (θ_D or θ_G) parameters fluctuate and are generally not stable during the training. The generator seldom achieves a point where it outputs very high-quality images.

3.1 Mode Collapse

Data distributions are multi-modal, meaning that every sample is classified (usually) to only one label. For example, in MNIST, there are ten classes of digits (or modes) labeled from "0" to "9." Mode collapse is the phenomenon where the generator only yields a small subset of the possible modes. In Fig. 2, you can observe generated MNIST images by two different GANs. The top row shows the training of a "good" GAN, not suffering from mode collapse. It generates every kind of mode (digit type) throughout the training. The bottom row exhibits a GAN training with mode collapse, generating just the digit "6."

3.2 Vanishing Gradients

GANs often suffer from training instability, where D performs very well and G does not get a chance to train a good distribution. We turn to provide some mathematical

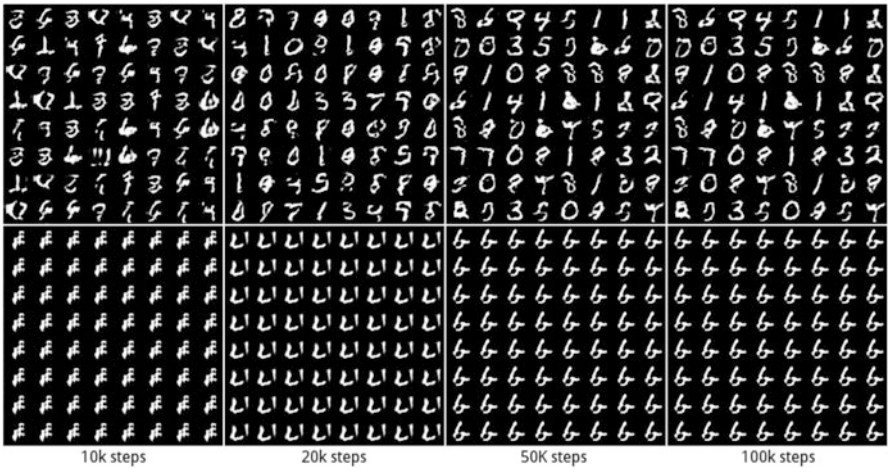


Fig. 2 Example of the mode collapse problem in GANs. The top row shows a training without mode collapse, where all MNIST modes (digits) are generated. The bottom row shows a bad training with mode collapse, where the generator outputs only the digit "6." The figure was taken from [44]

observations and understanding of why training a generator G is extremely hard when the discriminator D is close to optimal.

The global optimality, stated in [20], is defined when D is optimized for any given G. The optimal D is achieved when its derivative for Eq. (1) equals 0:

$$\begin{aligned} \frac{p_r(\mathbf{x})}{D(\mathbf{x})} - \frac{p_g(\mathbf{x})}{1 - D(\mathbf{x})} &= 0, \\ D^*(\mathbf{x}) &= \frac{p_r(\mathbf{x})}{p_r(\mathbf{x}) + p_g(\mathbf{x})}, \end{aligned} \tag{3}$$

where $\mathbf{x}$ is the real input data, $D^*(\mathbf{x})$ is the optimal discriminator, and $p_r(\mathbf{x})/p_g(\mathbf{x})$ is the distribution of the real/generated data, respectively, over the real data $\mathbf{x}$. If we substitute the optimal discriminator $D^*(\mathbf{x})$ into Eq. (1), we can visualize the loss for the generator G:

$$\mathbb{L}_G = \mathbb{E}_{\mathbf{x}\sim p_r} \log \frac{p_r(\mathbf{x})}{\frac{1}{2}[p_r(\mathbf{x}) + p_g(\mathbf{x})]} + \mathbb{E}_{\mathbf{x}\sim p_g} \log \frac{p_g(\mathbf{x})}{\frac{1}{2}[p_r(\mathbf{x}) + p_g(\mathbf{x})]} - 2 \cdot \log 2. \tag{4}$$

Before we continue any further, we mention two important metrics for probability measurement. The first one is the Kullback–Libeler (KL) divergence:

$$KL(p_1||p_2) = \mathbb{E}_{\mathbf{x}\sim p_1} \log \frac{p_1}{p_2}, \tag{5}$$

which measures how much the distribution p_2 differs from the distribution p_1. Note that this metric is not symmetrical, i.e., $KL(p_1||p_2) \neq KL(p_2||p_1)$. A symmetrical metric is the Jensen–Shannon (JS) divergence, defined as

$$JS(p_1||p_2) = \frac{1}{2} KL\left(p_1||\frac{p_1 + p_2}{2}\right) + \frac{1}{2} KL\left(p_2||\frac{p_1 + p_2}{2}\right). \tag{6}$$

Back to our GAN with the optimal D, Eq. (4) shows that the GAN loss function can be reformulated as

$$\mathbb{L}_G = 2 \cdot JS(p_r||p_g) - 2 \cdot \log 2, \tag{7}$$

which shows that for D^*, the generator loss turns to be a minimization of the JS divergence between p_r and p_g. The relation between GAN training and the JS divergence may explain its instability. To understand this, view Fig. 3, which shows an example for JS divergences of different distributions. In Fig. 3a, we see the real image distribution p_r, as a Gaussian with zero mean, and we consider three examples for generated image distributions: p_{g1}, p_{g2}, and p_{g3}. Figure 3(b) plots the $JS(p_r(\mathbf{x}), p_q(\mathbf{x}))$ measure between p_r and some p_q distribution, where the mean of p_q ranges from 0 to 80. As shown in the red box, the gradient of the JS

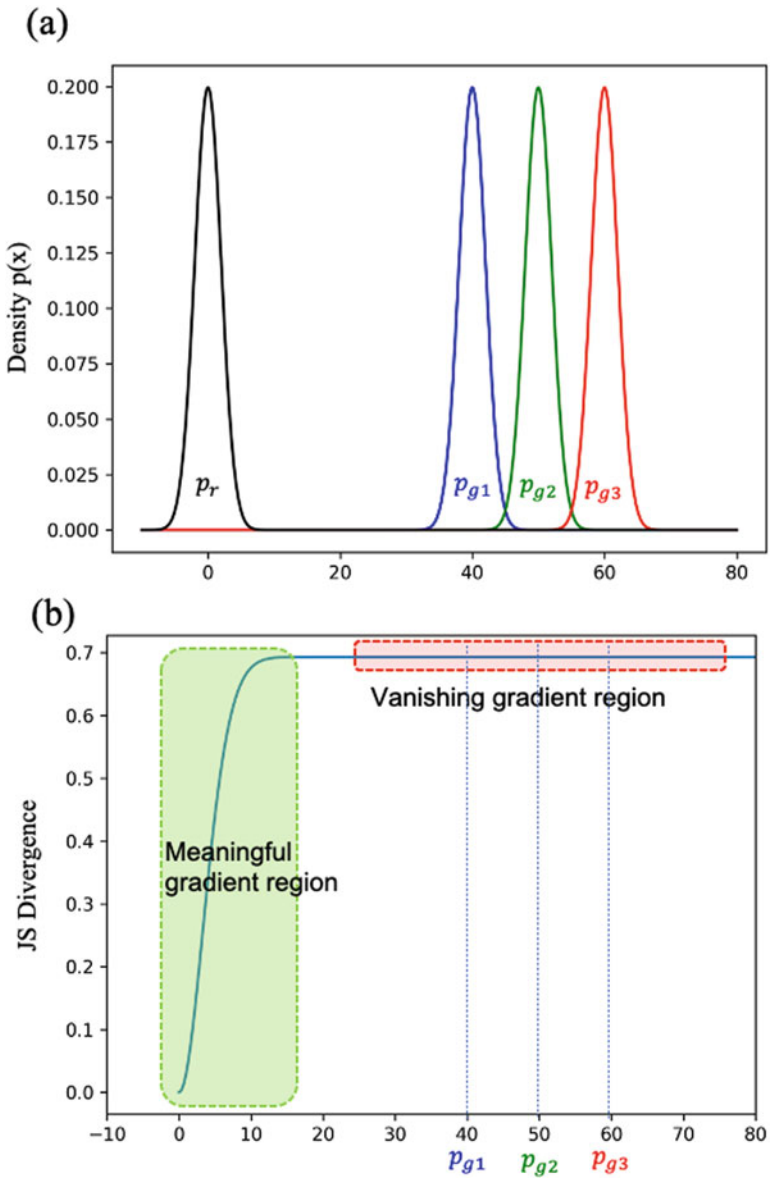


Fig. 3 Vanishing gradients in GANs. (**a**) An example of a real image distribution $p_r(\mathbf{x})$, a Gaussian with zero mean, with three more different Gaussian distributions: p_{g1}, p_{g2}, and p_{g3}. (**b**) Calculating the JS divergence measure between $p_r(\mathbf{x})$ and a Gaussian distribution with mean from 0 to 80. When training with optimal discriminator ($D^*(\mathbf{x})$), the generator G minimizes the loss in Eq. (6), pushing $p_g(\mathbf{x})$ left toward $p_r(\mathbf{x})$, alas, it could take a very long time due to diminished gradients when being far from $p_r(\mathbf{x})$

divergence vanishes after a mean of 30. In other words, when the discriminator is close to optimal ($D^*(\mathbf{x})$) and we try to train a "poor" generator with p_g far from the real distribution p_r, training will not be feasible due to extremely low gradients. This is prominent especially in the beginning of the training where G weights are randomized.

Due to the vanishing gradient problem, Goodfellow et al. [20] proposed a change to the original adversarial loss. Instead of minimizing $\log(1-D(G(\mathbf{z})))$ (as in Eq. 2), they suggest to maximize $\log(D(G(\mathbf{z})))$. The latter cost function yields the same fixed point of D and G dynamics but maintains higher gradients early in the training when the distributions p_r and p_g are far from each other. On the other hand, this training strategy promotes mode collapse, as we show next.

With an optimal discriminator D^*, $KL(p_g||p_r)$ can be reformulated as

$$\begin{aligned} KL(p_g||p_r) &= \mathbb{E}_{\mathbf{x}\sim p_g} \log \frac{p_g(x)/(p_r(x)+p_g(x))}{p_r(x)/(p_r(x)+p_g(x)))}, \\ &= \mathbb{E}_{\mathbf{x}\sim p_g} \log \frac{1-D^*(x)}{D^*(x)}, \\ &= \mathbb{E}_{\mathbf{x}\sim p_g} \log[1-D^*(x)] - \mathbb{E}_{\mathbf{x}\sim p_g} \log[D^*(x)]. \end{aligned} \tag{8}$$

If we switch the order of the two sides in Eq. (8), we get

$$\begin{aligned} -\mathbb{E}_{\mathbf{x}\sim p_g} \log[D^*(x)] &= KL(p_g||p_r) - \mathbb{E}_{\mathbf{x}\sim p_g} \log[1-D^*(x)], \\ &= KL(p_g||p_r) - 2\cdot JS(p_r||p_g) + 2\cdot\log 2 + \mathbb{E}_{\mathbf{x}\sim p_r} \log[D^*(x)]. \end{aligned} \tag{9}$$

The alternative loss for G is thus only affected by the first two terms (the last two terms are constant). Since $JS(p_r||p_g)$ is bounded by $[0, \log 2]$ (see Fig. 3b), the loss function is dominated by $KL(p_g||p_r)$, which is also called the reverse KL divergence. Since $KL(p_g||p_r)$ usually does not equal $KL(p_r||p_g)$, the optimized p_g by the reversed KL is totally different from p_g optimized by the KL divergence. Figure 4 shows the difference of the two optimizations where the distribution p is a mixture of two Gaussians, and q is a single Gaussian. When we optimize for $KL(p_r||p_g)$, q averages all of p modes to hit the mass center (Fig. 4a). However, for the reverse KL divergence optimization, q distribution chooses a single mode (Fig. 4b), which will cause a mode collapse during training.

In summary, using the original G loss from Eq. (1) will result in vanishing gradients for G, and using the other loss in Eq. (9) will result in a mode collapse. These problems are inherent within the GAN loss and thus cannot be solved using sophisticated architectures. In Sect. 5, we discuss other loss functions for GANs, which solve these problems.

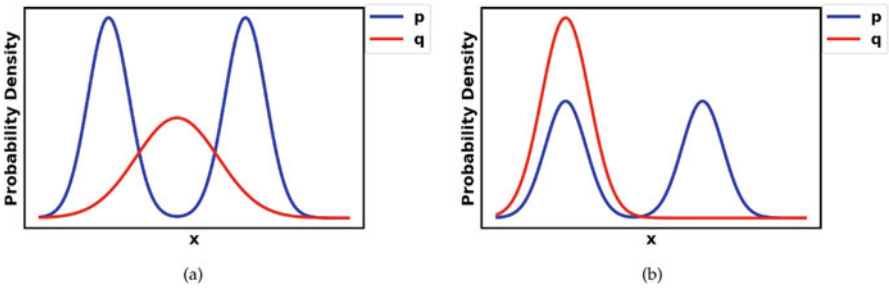


Fig. 4 Optimized distribution q for (**a**) minimizing KL divergence $KL(p||q)$ and (**b**) minimizing reverse KL divergence $KL(q||p)$. Figure is taken from [71]

3.3 Instability and Image Quality

Early GAN models that use the G losses described above $\left(\log\left(1 - D(G(\mathbf{z}))\right)\right.$ and $\left. -\log\left(D(G(\mathbf{z}))\right)\right)$ exhibit great instability in their cost values during training. Arjovsky et al. [4] experimented with these two G losses and claimed that in both cases the gradients cause instability to the GAN training. The loss in Eq. (1) stays constant after the first training steps (Fig. 5a), and the loss in Eq. (8) fluctuates during the entire training (Fig. 5b). In both cases, they did not find a significant correlation between the calculated loss and the generated image quality. In other words, it is very difficult to predict when during the training the generator actually produces good quality images, and the only way to get a good generator is to stop the training and manually visualize many generated images.

Using the above two G losses in Eq. (1) and Eq. (8) yields poor quality images compared to modern GAN models. In the following sections, we will cover more sophisticated losses that enhance the generator's resolution and image size.

3.4 Problems: Summary

The original GAN model and loss proposed by Goodfellow et al. [20] suffer from three inherent challenges: (1) mode collapse, (2) vanishing gradients, and (3) image quality. Follow-up works improve the performance of GANs on one or more of these problems by using different architectures for D or G, modifying the cost function, and more. A subset of architecture-variant and loss-variant GANs are portrayed in Fig. 6a,b, respectively. A sample of some recent prominent GAN models is presented in the following sections.

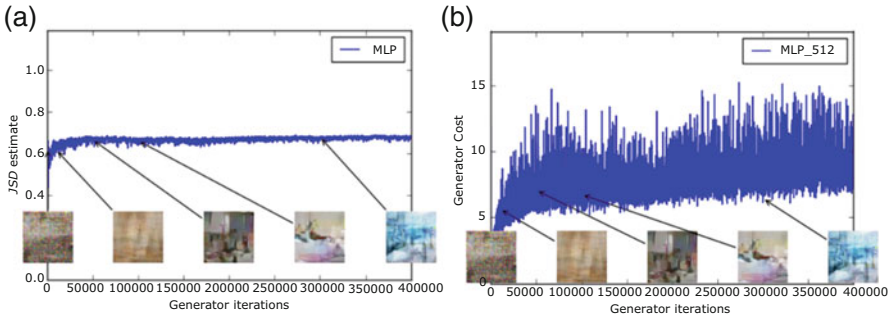


Fig. 5 Training instability in GANs. (**a**) JS distance metric in GAN training using the loss in Eq. (1). This quantity correlates poorly to the generated image quality, saturating to $\log 2 \approx 0.69$, the highest value taken by the JS distance. (**b**) Generator loss during training using a different generator cost (maximizing $\log(D(G(\mathbf{z})))$ instead of minimizing $\log(1 - D(G(\mathbf{z})))$) showing increasing error without a significant improvement in image quality. Plots are taken from [4]

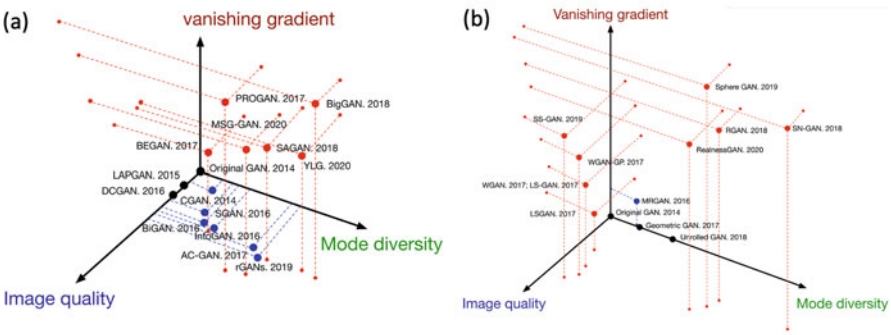


Fig. 6 Recent GAN models that solve the original GAN's problems: mode diversity (collapse), vanishing gradients, and image quality. (**a**) A subset of architecture-variant GANs (**b**) a subset of loss-variant GANs. Larger axis values indicate better performance. Red points indicate the model improves all three challenges, blue points improve two, and black points improve only one. Figures are taken from [72]

4 Improved GAN Architectures

Many types of new GAN architectures have been proposed since 2014 [6, 8, 13, 29, 57, 77, 30, 16]. Different GAN architectures were proposed for different tasks, such as image super-resolution [38] and image-to-image transfer [79, 53]. In this section, we present some of these models, which improved the performance on image quality, vanishing gradients, and mode collapse, compared to the original GAN.

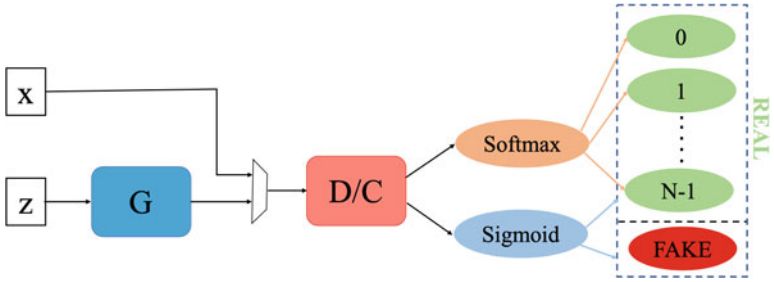


Fig. 7 SGAN architecture

4.1 Semi-Supervised GAN (SGAN)

Semi-supervised learning is a promising research field between supervised learning and unsupervised learning. In supervised learning, each data is labeled, and in unsupervised learning, no labels are provided; semi-supervised learning has labels only for a small subset of the training data and no annotations for the rest of the data, like many real-world problems.

SGAN [49] extended the original GAN learning to the semi-supervised context by adding to the discriminator network an additional task of classifying the image labels (Fig. 7). The generator's architecture is the same as before. In SGAN, the discriminator utilizes two heads: a softmax and a sigmoid. The sigmoid is used to distinguish between real and fake images, and the softmax predicts the images' labels (only for images predicted as real). The results on MNIST showed that both D and G in SGAN are improved compared to the original GAN.

4.2 Conditional GAN (CGAN)

CGAN was originally proposed as an extension of the original GAN, where both the discriminator and the generator were fed by an additional class of the image [45, 50]. An illustration of CGAN architecture is shown in Fig. 8. In this setup, the loss function in Eq. (1) is slightly modified to condition both the real images x and the latent variable z on y (the label):

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_r} \log[D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}} \log[1 - D(G(\mathbf{z}|\mathbf{y}))]. \tag{10}$$

All values (x, y, z) are first encoded by some neural layers prior to their fusion in the discriminator and generator. This improves the ability of the discriminator to classify real/fake images and enhances the generator's ability to control the modalities of the generated images. Reference [45] showed that their CGAN architecture used with a language model can handle also multimodal datasets, such

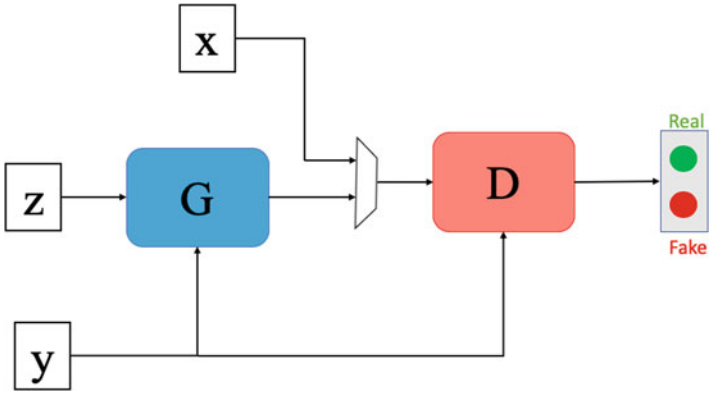


Fig. 8 CGAN architecture. Both the generator and the discriminator are fed with a class label y to condition both the images x and the latent variable z (see Eq. 10)

as Flicker that contains labeled image data with their particular user tags. They demonstrated that their generator is capable of producing an automatic tagging.

4.3 Deep Convolutional GAN (DCGAN)

Before describing DCGAN, we provide a brief reminder of what is a convolutional neural network.

CNNs: Convolutional neural networks (CNNs) were proposed by LeCun et al. [37]; These networks consist of trained spatial filters applied on hidden activations throughout their architecture. these networks perform correlations using their trained kernels with a sliding window over images (or hidden activations). This was shown to improve the accuracy on many recognition tasks, especially in computer vision. A very basic and popular CNN network called LeNet is shown in Fig. 9.

DCGAN proposed to create images using solely deconvolutional networks in their generator [57]. The generator architecture is depicted in Fig. 10. Deconvolutional networks can be conceived as CNNs that use the same components but in reverse, projecting features into the image pixel space. Reference [76] showed that deconvolutional layers achieve good visualization for CNNs; this allowed the DCGAN generator to create high-resolution images for the first time.

In addition to improved resolution, DCGAN showed better stability during the training thanks to multiple modifications in the original GAN network:

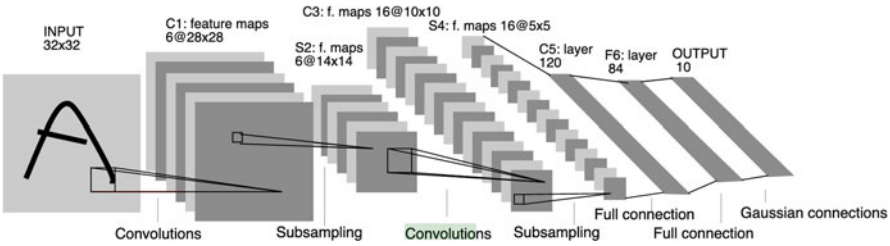


Fig. 9 LeNet architecture. A popular Convolutional Neural Network (CNN) used for image recognition. Image was taken from [37]

- All pooling layers were replaced. In the discriminator, they used convolution kernels with stride > 1, and the generator utilized fractionally strided convolution to increase the spatial size.
- Both the discriminator and the generator were trained with batch normalization which promotes similar statistics for real images and fake generated images.
- The discriminator architecture replaces all normal ReLU activations with Leaky-ReLU [40]; this activation multiplies also the negative kernel output by a small value (0.2), to prevent from "dead" gradients to propagate to the generator. The generator architecture used ReLU after every deconvolution layer except the output, which uses Tanh activation.

The authors demonstrated empirically that the mere CNN architecture used in DCGAN is not the key contributing factor for the GAN's performance, and the above modifications are crucial. To show that they measured GAN quality by considering them as feature extractors on supervised datasets and evaluating the performance of linear models trained on these features (for more information, see [57]). DCGAN yields a classification error of 22.48% on the StreetView House Numbers (SVHN) dataset [47], whereas a purely supervised CNN with the same architecture achieved a significantly higher 28.87% test error.

4.4 Progressive GAN (PROGAN)

PROGAN described a novel training methodology for GANs, involving progressive steps toward the development of the entire network architecture [29]. This progressive architecture uses the idea of progressive neural networks first proposed by Rusu et al. [62]. These architectures are immune to forgetting and can leverage prior knowledge via lateral connections to previously learned features. The progressive training scheme is shown in Fig. 11. First, they trained low resolution $4x4$ pixel images. Next, both D and G grow to include a layer of $8x8$ spatial resolution. This progressive training gradually adds more and more intermediate layers to enhance

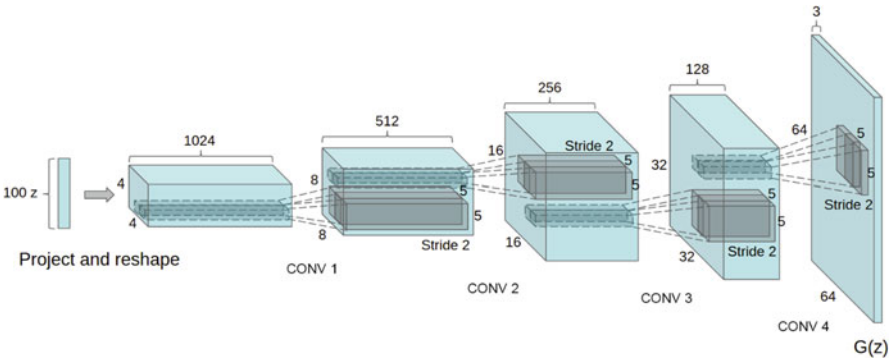


Fig. 10 DCGAN generator architecture. This generator creates a complex 64 × 64 pixel image from a 100-dimensional uniform distribution z. No fully connected or pooling layers are used. Figure is taken from [57]

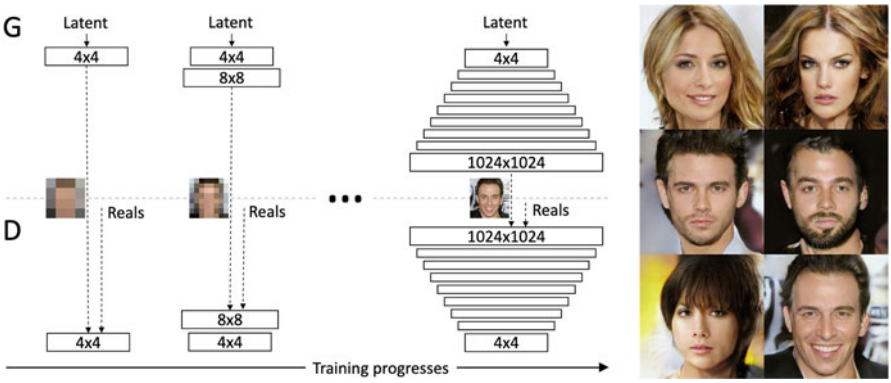


Fig. 11 Training process for the PROGAN progressive methodology. Training starts with both G and D having a low spatial resolution of 4 × 4 pixels. Every training step adds a new incremented intermediate layer to D and G, enhancing the generated images' resolution. All existing layers are trainable throughout the process. The images on the right are examples of generated images using PROGAN at 1024 × 1024. Figure was taken from [29]

the resolution, until reaching a high resolution of 1024 × 1024 pixel images with the CelebA dataset. All previous layers remain trainable in later steps.

Many state-of-the-art GAN architectures utilize this type of progressive training scheme, and it has resulted in very credible images [29, 30, 8, 32, 60] and more stable learning for both D and G.

Fig. 12 Using truncated sampling from the latent space. (**a**) Trade-off between image quality and image variety. From left to right, the truncation threshold is set to 2, 1, 0.5, and 0.04. (**b**) Saturation artifacts from applying truncated normal distribution to a model training with $z \sim \mathcal{N}(0,\ 1)$. Images were taken from [8]

4.5 BigGAN

Attention in computer vision is a method that focuses the task on an "interesting" region in the image. The self-attention is used to train the network (usually CNN) in an unsupervised manner, teaching it to segment or localize the most relevant pixels (or activations) for the vision task.

BigGAN [8] has achieved state-of-the-art generation on the ImageNet datasets. Its architecture is based on the Self-attention GAN (SAGAN) [77], which employs a self-attention mechanism in both D and G, to capture a large receptive field without sacrificing computational efficiency for CNNs [69]. The original SAGAN architecture can learn global semantics and long-range dependencies for images, thus generating excellent multi-label images based on the ImageNet datasets (128×128 pixels). BigGAN achieved improved performance by scaling up the GAN training: increasing the number of network parameters ($\times 4$) and increasing the batch size ($\times 8$). They achieved better performance on ImageNet with size 128×128 and were also able to train BigGAN also on the resolutions 256×256 and 512×512.

Unlike many previous GAN models, which randomized the latent variable from either $z \sim \mathcal{N}(0,\ 1)$ or $z \sim \mathcal{U}(-1,\ 1)$, BigGAN uses a simple *truncation trick*. During training, z is sampled from $\mathcal{N}(0,\ 1)$, but for generating images in inference z is selected from a truncated normal, where values that lie outside the range are resampled until falling in the range. This *truncation trick* shows improvement in individual sample quality at the cost of a reduction in overall sample variety, as shown in Fig. 12a. Notice though that using this different inference sampling trick as is may not generate good images with some large models, producing some saturation artifacts (Fig. 12b). To solve this issue, the authors proposed to use Orthogonal Regularization to force G to be more amenable to truncation, making it smoother so that the full lateral space will map to good, generated images.

4.6 StyleGAN

StyleGAN [30] proposed an alternative generator architecture for GANs. Unlike the traditional G architecture that samples a random latent variable at its input, their architecture starts from a learned constant input and adjusts the characteristics of the images in every convolutional layer along G using different, learned latent variables. Additionally, they inject learned noise directly throughout G (see Fig. 13).

This architectural change leads to automatic, unsupervised separation of high-level attributes (e.g., pose and identity) from low-level variations (e.g., freckles, hair) in the generated images. StyleGAN did not modify the discriminator or the loss function. They used the same D architecture as in [29]. In addition to state-of-the-art quality for face image generation, StyleGAN demonstrates a higher degree of latent space disentanglement, presenting more linear representations of different factors of variation, turning the GAN synthesis to be much more controllable. Recently, a more advanced styleGAN architecture has been proposed [32, 18, 33]. Some generated examples are presented in Fig. 14. One may also use StyleGAN for image editing by

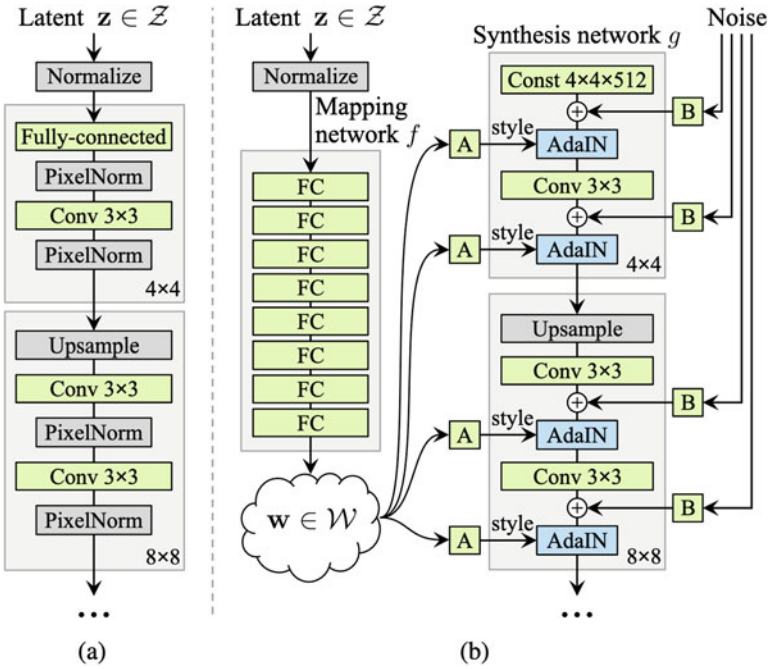


Fig. 13 Comparison between a traditional generator (**a**) and the StyleGAN generator (**b**). Unlike the standard generator that selects a random latent variable only in its input, in StyleGAN generator the latent input is mapped to an intermediate latent space $\mathcal{W}$, which then controls the generator through adaptive instance normalization (AdaIN). Also, Gaussian noise is injected after every convolution layer. "A" corresponds to a learned affine transform, and "B" applies learned scaling factors to the input noise. Figure is taken from [30]

Fig. 14 Selected face images generated using the StyleGAN2 architecture (an improved version of the original StyleGAN), trained on the FFHQ dataset. Figure is taken from [32]

calculating the latent vector of a given input image in the styleGAN (this operation is known as styleGAN inversion) and then manipulating this vector for editing the image [75, 60, 1, 68, 2, 65, 64, 54].

5 Improved GAN Objectives

As described in Sect. 3.2, the original GAN's min–max loss (Eq. 1) promotes mode collapse and vanishing gradient phenomena. This section presents selected loss functions and regularizations that remedy these problems and also improves the image quality. This is just a partial list, and other loss functions and regularization methods exist such as the least square GANs [41] or optimal transport models [55].

5.1 Wasserstein GAN (WGAN)

WGAN [4] has solved the vanishing gradient and mode collapse problems of the original GAN by replacing the cost in Eq. (1) with the Earth Mover (EM) distance, which is known also as the Wasserstein distance and is defined as

$$W(p_r, p_g) = \inf_{\gamma \in \prod(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||], \tag{11}$$

where $\prod(p_r, p_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are p_r and p_g, respectively. The EM distance is therefore the minimum cost of transporting "mass" in converting distribution p_r into the distribution p_g.

Unlike KL and JS distances, EM is capable of indicating distance even when the p_r and p_g distributions are far from each other; EM is also continuous and thus provides useful gradients for training G. However, the infimum in Eq. (11) is highly intractable, so the authors estimate the EM cost with

$$\max_{w\sim W} \mathbb{E}_{\mathbf{x}\sim p_r}[f_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_z}[f_w(G(\mathbf{z}))], \tag{12}$$

where $\{f_w\}_{w\in\mathcal{W}}$ is a parameterized family of all functions that are K-Lipschitz for some K ($||f||_L \leq K$). The readers are referred to [4] for more details.

The authors proposed to find the best function f_w that maximizes Eq. (12) by back-propagating $\mathbb{E}_{\mathbf{z}\sim p_z}\left[\nabla_\theta f\left(g_\theta(z)\right)\right]$, where g_θ are the generator's weights. f_w can be realized by D but constrained to be K-Lipschitz and z is the lateral input noise for G. w in f_w are the discriminator's parameters, and the objective of D is to maximize Eq. (12), which approximates the EM distance. When D is optimized, Eq. (12) becomes the EM distance, and G is optimized to minimize it:

$$-\min_G \mathbb{E}_{\mathbf{z}\sim p_z}[f_w(G(\mathbf{z}))]. \tag{13}$$

Figure 15 compares the gradient of WGAN to the original GAN from two non-overlapping Gaussian distributions. It can be observed that WGAN has a smooth and measurable gradient everywhere (blue line) and learns better even when G is not producing good images.

5.2 Self-Supervised GAN (SSGAN)

We showed in Sect. 4.2 that CGANs can generate natural images. However, they require labeled images to do so, which is a major drawback. SSGANs [10] exploit two popular unsupervised learning techniques, adversarial training, and self-supervision, bridging the gap between conditional and unconditional GANs.

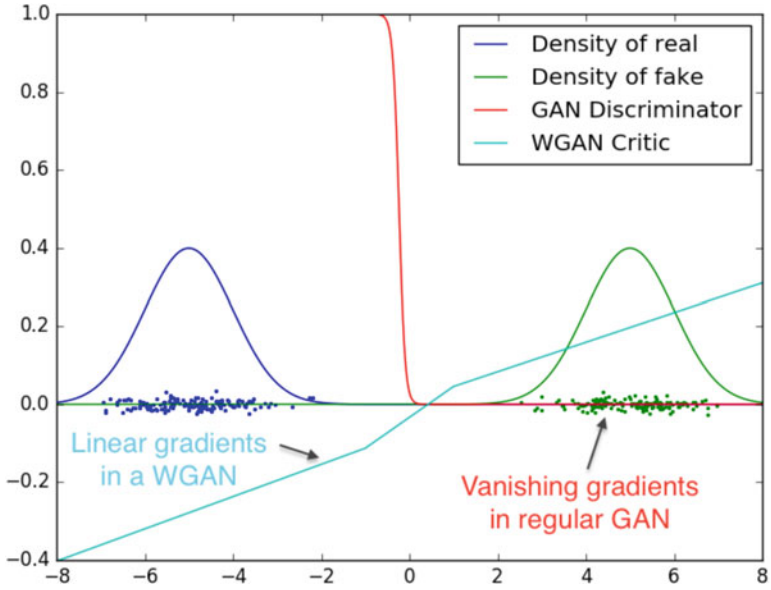


Fig. 15 Differentiating two Gaussian distributions using optimal discriminator D^*. The original GAN cost saturates (red line), resulting in vanishing gradients, whereas WGAN cost objective (blue line) yields measurable gradients. Plot was taken from [4]

Neural networks have been shown to forget previous learned tasks [17, 35], and catastrophic forgetting was previously considered as a major cause for GAN training instability. Motivated by the desire to counter the discriminator forgetting, SSGAN adds to the discriminator an additional loss, which enables it to learn useful representations, independently of the quality of the generator. In a self-supervised manner, the authors train a model on a task of predicting a rotation angle ($[0°, 90°, 180°, 270°]$), as shown in Fig. 16. The objectives of D and G are updated to

$$\begin{aligned} L_G &= -V(G,D) - \alpha \mathbb{E}_{\mathbf{x}\sim P_G} \mathbb{E}_{r\sim\mathbb{R}}[\log Q_D(R=r|\mathbf{x}^r)] \\ L_D &= V(G,D) - \beta \mathbb{E}_{\mathbf{x}\sim P_{data}} \mathbb{E}_{r\sim\mathbb{R}}[\log Q_D(R=r|\mathbf{x}^r)], \end{aligned} \tag{14}$$

where $V(G, D)$ is the original GAN objective in Eq. (1), P_{data} and P_G are the real data and generated data distributions, respectively, and $r \in R$ is a rotation selected from a set of all allowed angles ($R = \{0°, 90°, 180°, 270°\}$). An image $\mathbf{x}$ rotated by r degrees is denoted as $\mathbf{x}^r$, and $Q(R|\mathbf{x}^r)$ is the discriminator's predictive distribution over the angles of rotation of the sample. These new losses enforce D to learn good representation via learning the rotation information in a self-supervised approach.

Using the above scheme, SSGAN achieves good high-quality images, matching the performance of conditional GANs without having access to labeled data.

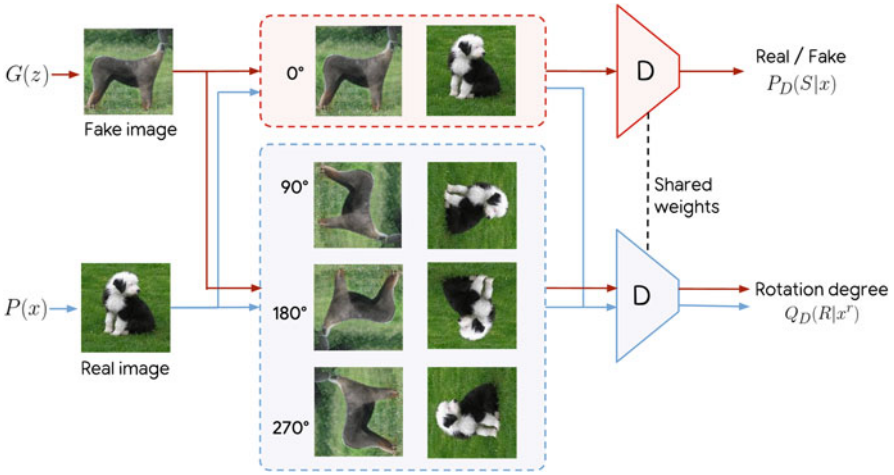


Fig. 16 Self-Supervised GAN (SSGAN) discriminator. The discriminator, D, performs two tasks: identifying real/fake images (as the original GAN) and rotation classification. Both the real and fake images are rotated by ($[0°, 90°, 180°, 270°]$) and sent to the rotation degree classifier (in blue). Only the original 0° images are sent to the real/fake classifier (in red). Plot is taken from [10]

5.3 Spectral Normalization GAN (SNGAN)

SNGAN [46] proposes to add weight normalization to stabilize the training of the discriminator. Their technique is computationally inexpensive and can be applied easily to existing GANs architectures. In previous works that stabilized GAN training [21, 4, 56], it was emphasized that D should be a K-Lipschitz continuous function, forcing it not to change rapidly. This characteristic of D stabilizes the training of GANs. SNGAN controls the Lipschitz constant of D by literally constraining the spectral norm of each layer, normalizing each weight matrix W, so it satisfies the spectral constraint $\sigma(\mathbf{W}) = 1$ (i.e., the largest singular value of the weight matrix of each layer is 1). This is performed by simply normalizing each layer:

$$\bar{\mathbf{W}}_{SN}(\mathbf{W}) = \frac{\mathbf{W}}{\sigma(\mathbf{W})}, \tag{15}$$

where $\mathbf{W}$ are the weight parameters of each layer in D. This work proves that this will make the Lipschitz constant of the discriminator function to be bound by 1, which is important for the WGAN optimization.

SNGAN achieves an extraordinary advance on ImageNet, and better or equal quality on CIFAR-10 and STL-10, compared to the previous training stabilization techniques that include weight clipping [4], gradient penalty [74, 43], batch normal-

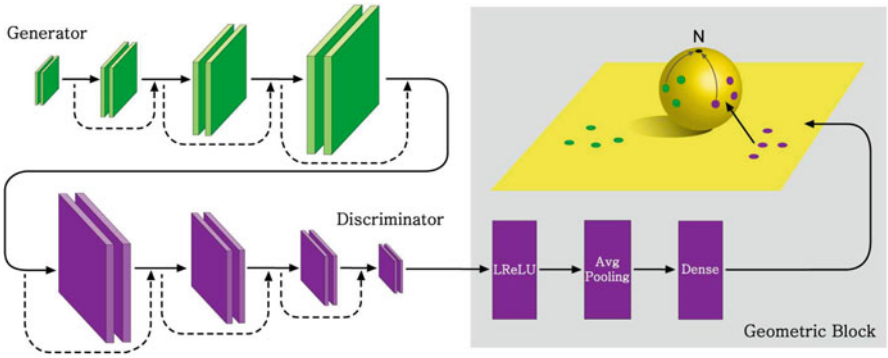


Fig. 17 Pipeline of Sphere GAN. The generator creates fake data from noise inputs. Real and fake data are fed to the discriminator, which maps the output to an n-dimensional Euclidean feature space (yellow plane). Green and purple points on the feature plane correspond to fake and real samples, respectively. The key idea of SphereGAN is remapping the feature points into the n-dimensional hypersphere by using geometric transformations. These mapped points are then used to calculate the geometric moments centered at the north pole of the hypersphere (N). While the discriminator tries to maximize the moment differences of real and fake samples, the generator tries to interfere with the discriminator by minimizing those moment differences. Figure was taken from [52]

ization [26], weight normalization [63], layer normalization [5], and orthonormal regularization [7].

5.4 SphereGAN

SphereGAN [52] is a novel integral probability metric (IPM)-based GAN, which uses the hypersphere to bound IPMs in the objective function, thus enhancing the stability of the training. By exploiting the information of higher order statistics of data using geometric moment matching, they achieved more accurate results. The objective function of SphereGAN is defined as

$$\min_G \max_D \sum_r E_x[d_s^r(\mathbf{N}, D(x))] - \sum_r E_z[d_s^r(\mathbf{N}, D(G(z)))], \tag{16}$$

for $r = 1, \ldots, R$ where the function d_s^r measures the rth moment distance between each sample and the north pole of the hypersphere, $\mathbf{N}$. Note that the subscript s indicates that d_s^r is defined on $\mathbb{S}^n$. Figure 17 shows the pipeline of SphereGAN. By defining IPMs on the hypersphere, SphereGAN can alleviate several constraints that should be imposed on D for stable training, such as the Lipschitz constraints required from conventional discriminators based on the Wasserstein distance.

Unlike conventional approaches that use the Wasserstein distance and add additional constraint terms (see Table 1 in [52]), SphereGAN does not need any additional constraints to force D in a desired function space, due to the usage of geometric transformation in D.

6 Data Augmentation with GAN

We turn now to describe how to use GAN for data augmentation. Data augmentation is a crucial process for tasks lacking large training sets. There are three circumstances which require data augmentation:

- **Limited annotations**: Training a large DNN with a very few labeled data in the training set.
- **Limited diversity**: When the training data lacks variations; for example, it may not cover diverse illuminations or a variety of appearances.
- **Restricted data**: The database might contain sensitive information, and thus accessing it directly is strictly restricted.

The first two scenarios can be solved via supervised learning, but they will cost a lot of human effort to enrich the labeled data or to use active learning approaches [11]. An alternative approach is to utilize GANs to augment the data. Semi-supervised GAN (SGAN) (see Sect. 4.1) is a simple example of such a GAN architecture that is able to generate new annotated images and can automatically enrich training data with few labels.

The second case is more common in the real world. Data variability is important for many psychology or neuroscience experiments. For example, a human's EEG is sensitive to different types of face images such as happy, angry, and sad faces [42]. Preparing those stimuli in traditional experiments is time-consuming and costly for researchers. Architectures like StyleGAN [30] (see Sect. 4.6) are specialized to generate broad types of face images as a stimulus. More important, these images can be controlled to exhibit specific attributes (e.g., level of happiness or facial textures) and thus enhancing the stimuli variation in the experiment [73]. Data augmentation can also be helpful in the training of the GANs themselves [31, 78], which allows to train them with only few examples and still get high-quality generated data (e.g., images).

The last case is related to unsupervised learning approaches. When a database is restricted to preserve users' privacy, the researchers can use GANs to synthesize this sensitive data themselves. For example, Delaney et al. employed GANs to generate synthetic ECG signals that resemble real ECG data [12].

7 Conclusion

This chapter has provided just a glimpse of GANs and their various usages. This fast-growing technique has many variants and applications that have not been presented here for the sake of brevity. Yet, we believe that the description of GANs given here provides the reader with the tools to better understand this tool and be able to navigate between the vast amount of recent literature on this topic. It is worth mentioning also normalizing flows [51] and score-based generative models [67, 66, 48] which become very popular recently and show competitive performance with GANs.

Acknowledgements We would like to thank Yuval Alaluf, Yotam Nitzan, and Ron Mokady for their helpful comments.

References

1. Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2StyleGAN: How to embed images into the StyleGAN latent space? In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4431–4440, 2019.
2. Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2StyleGAN++: How to edit the embedded images? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
3. Rameen Abdal, Peihao Zhu, Niloy J. Mitra, and Peter Wonka. StyleFlow: Attribute-conditioned exploration of StyleGAN-generated images using conditional continuous normalizing flows. *ACM Trans. Graph.*, 40(3), May 2021.
4. Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *ICML*, volume 70, pages 214–223, 2017.
5. Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. 7 2016.
6. David Berthelot, Tom Schumm, and Luke Metz. {BEGAN:} Boundary Equilibrium Generative Adversarial Networks. *CoRR*, abs/1703.1, 2017.
7. Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural Photo Editing with Introspective Adversarial Networks. *ArXiv*, abs/1609.0, 2017.
8. Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.
9. Eoin Brophy, Zhengwei Wang, and Tomas E. Ward. Quick and easy time series generation with established image-based GANs. *ArXiv*, abs/1902.05624, 2019.
10. Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-Supervised GANs via Auxiliary Rotation Loss. *CVPR*, pages 12146–12155, 2019.
11. David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active Learning with Statistical Models. *J. Artif. Int. Res.*, 4(1):129–145, 3 1996. ISSN 1076-9757.
12. Anne Marie Delaney, Eoin Brophy, and Tomas E Ward. Synthesis of Realistic ECG using Generative Adversarial Networks. *ArXiv*, abs/1909.0, 2019.
13. Emily L Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. *CoRR*, abs/1506.0, 2015.
14. Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, page 258–267, 2015.

15. William Fedus, Ian Goodfellow, and Andrew M. Dai. MaskGAN: Better text generation via filling in the _. In *International Conference on Learning Representations*, 2018.
16. Yuri Feigin, Hedva Spitzer, and Raja Giryes. Generative adversarial encoder learning, 2020.
17. Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.
18. Rinon Gal, Dana Cohen Hochberg, Amit Bermano, and Daniel Cohen-Or. SWAGAN: A style-based wavelet-driven generative model. *ACM Trans. Graph.*, 40(4), July 2021a.
19. Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. StyleGAN-NADA: Clip-guided domain adaptation of image generators, 2021b.
20. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680, 2014.
21. Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems 30*, pages 5767–5777, 2017.
22. Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5141–5148, 2018.
23. Kay Gregor Hartmann, Robin Tibor Schirrmeister, and Tonio Ball. EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals. *ArXiv*, abs/1806.01875, 2018.
24. Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In *International Conference on Machine Learning*, volume 80, pages 1989–1998, 2018.
25. Shady Abu Hussein, Tom Tirer, and Raja Giryes. Image-adaptive GAN based reconstruction. In *AAAI*, 2020.
26. Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 2015.
27. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2016.
28. Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *CoRR*, abs/1611.08207, 2016.
29. Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.
30. Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019.
31. Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Proc. NeurIPS*, 2020a.
32. Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020b.
33. Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Proc. NeurIPS*, 2021.
34. Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.
35. James N Kirkpatrick, Razvan Pascanu, Neil C Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114:3521–3526, 2017.
36. Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic DENOYER, et al. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, pages 5963–5972, 2017.

37. Y LeCun, P Haffner, L Bottou, and Yoshua Bengio. Object Recognition with Gradient-Based Learning. In *Shape, Contour and Grouping in Computer Vision*, 1999.
38. Christian Ledig, Lucas Theis, Ferenc Huszár, José Antonio Caballero, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017.
39. Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. Semantic segmentation using adversarial networks. *ArXiv*, abs/1611.08408, 2016.
40. Andrew L Maas. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *ICML*, 2013.
41. Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
42. Aimee Mavratzakis, Cornelia Herbert, and Peter Walla. Emotional facial expressions evoke faster orienting responses, but weaker emotional responses at neural and behavioural levels compared to scenes: A simultaneous EEG and facial EMG study. *NeuroImage*, 124:931–946, 2016.
43. Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for GANs do actually converge? In *International Conference on Machine Learning (ICML)*, 2018.
44. Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017.
45. Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *ArXiv*, abs/1411.1, 2014.
46. Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations*, 2018.
47. Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
48. Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, volume 139, pages 8162–8171, 2021.
49. Augustus Odena. Semi-Supervised Learning with Generative Adversarial Networks. *ArXiv*, abs/1606.0, 2016.
50. Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis with Auxiliary Classifier {GAN}s. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2642–2651, 2017.
51. George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.
52. Sung Woo Park and Junseok Kwon. Sphere Generative Adversarial Network Based on Geometric Moment Matching. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4287–4296, 2019.
53. Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
54. Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. StyleCLIP: Text-driven manipulation of StyleGAN imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2085–2094, October 2021.
55. Gabriel Peyre and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.
56. Guo-Jun Qi. Loss-Sensitive Generative Adversarial Networks on Lipschitz Densities. *International Journal of Computer Vision*, 128:1118–1140, 2019.

57. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR*, abs/1511.0, 2015.
58. Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.
59. Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.
60. Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a StyleGAN encoder for image-to-image translation. In *CVPR*, 2021.
61. Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. SinGAN: Learning a generative model from a single natural image. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.
62. Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *ArXiv*, abs/1606.0, 2016.
63. Tim Salimans and Durk P Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems 29*, pages 901–909. Curran Associates, Inc., 2016.
64. Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in GANs. In *CVPR*, 2021.
65. Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. InterfaceGAN: Interpreting the disentangled face representation learned by GANs. *TPAMI*, 2020.
66. Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.
67. Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.
68. Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for StyleGAN image manipulation. *ACM Trans. Graph.*, 40(4), 2021.
69. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.
70. T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, June 2018. doi:10.1109/CVPR.2018.00917.
71. Zhengwei Wang, Qi She, and Tomas E Ward. Generative Adversarial Networks: {A} Survey and Taxonomy. *CoRR*, abs/1906.0, 2019a.
72. Zhengwei Wang, Qi She, and Tomas E Ward. Generative Adversarial Networks: {A} Survey and Taxonomy. *CoRR*, abs/1906.0, 2019b.
73. Zhengwei Wang, Qi She, Alan F Smeaton, Tomás E Ward, and Graham Healy. Synthetic-Neuroscore: Using a neuro-AI interface for evaluating generative adversarial networks. *Neurocomputing*, 405:26–36, 2020.
74. Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. GP-GAN: Towards Realistic High-Resolution Image Blending. *Proceedings of the 27th ACM International Conference on Multimedia*, 2019.
75. Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey, 2021.
76. Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *ECCV*, 2014.

77. Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, volume 97, pages 7354–7363, 2019.
78. Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient GAN training. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
79. Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

Spatial Data Science

Yan Li, Yiqun Xie, and Shashi Shekhar

1 Introduction

Space and time are the context of observations in a large number of domains, such as climate science, social science, epidemiology, transportation, and criminology [8], so in these domains the spatial and temporal information of every measurement is often recorded in the observation data, hereby referred to as spatial data. For example, GPS tracks of smart phones keep track of the status of smart phone users as well as their geographic locations and timestamps. Remote sensing imagery is another example of spatial data. Each pixel of a remote sensing image reflects an attribute of the features at a location at the moment when the image is taken. In recent years, the explosive growth in spatial data has facilitated the emergence and development of spatial data science, which is a multi-disciplinary field that applies scientific methods from computer science, statistics, mathematics, and other domains acquire, store, and manage spatial data, as well as to retrieve previously unknown, but potentially useful and non-trivial knowledge and insights from the data. The life cycle of spatial data science has five phases, namely spatial data acquisition, spatial data storage and preprocessing, spatial data mining, result validation, and domain interpretation.

Spatial data science is crucial to applications that decide actions and policies on large spatial datasets in many domains, including national security [57], public health [56], transportation [1], and public safety [82]. The examples of the applications and domains are shown in Table 1. For instance, spatial data science applies spatial scan statistics to identify areas with significantly high concentration of disease from data of disease incidents to manage epidemic outbreak.

Y. Li · Y. Xie · S. Shekhar (✉)
University of Minnesota, Minneapolis, MN, USA
e-mail: lixx4266@umn.edu; xiexx347@umn.edu; shekhar@umn.edu

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_18

Table 1 Application examples of spatial data science

Domain	Application examples
Public safety	Detecting crime hotspots with concentrations of crime
Epidemiology	Identifying epidemic outbreak
Business	Allocating retail store locations to maximize profit
Neuroscience	Detecting brain regions activated by language from neuro-images
Climate science	Finding distant places where the temperature is positively or negatively correlated

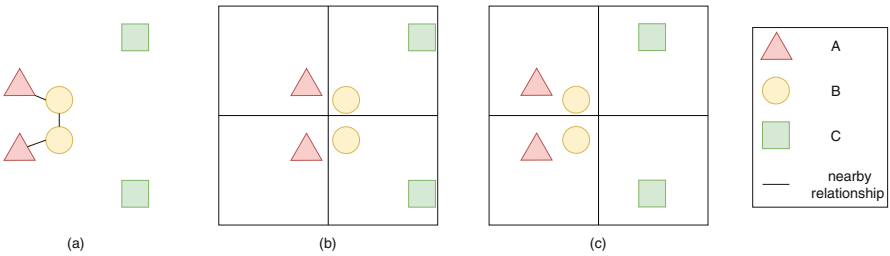


Fig. 1 Example of the modifiable area unit problem

One of the main challenges of spatial data science is its interdisciplinary nature. Spatial data science deals with the objects or phenomena that physically exist in the real world, so its techniques must be developed with an awareness of the underlying physics or theory in the application domain, and its results must be interpretable and reliable [53]. For example, climate science studies find that observable predictors for phenomena discovered by data science techniques can be misleading if they ignore climate models, locations, and seasons [13]. When simulating lake temperature profiles, by considering energy conservation, pre-training with physical simulation, and applying density-depth constraint, a physics-guided spatial data science model reduces physically inconsistent results and improves generalizability compared with the state-of-the-art non-spatial models [40]. Another good example is the Google Flu Trends, a project which attempted to predict flu activity only by tracking people's Google search about the illness. Despite its early success over the first several years, it overestimated the prevalence of flu in 2011 and 2012 by more than 50% [80]. False positive results from spatial data science may have profound negative impacts on the economy, society, and the environment. For example, labeling a neighborhood as a crime hotspot may reduce property values in the neighborhood [46]. It is critical to have robust statistical significance testing that can further validate or discard the knowledge obtained from spatial data.

The complexity of spatial data types and relationships also makes extracting knowledge from spatial data more difficult than from non-spatial numeric and categorical data. Commonly used spatial data types include object data types, such as points, lines, and polygons, and field data types such as remote sensing images and digital elevation models, which is more complex than those in non-spatial data

science. In addition, some important relationships in non-spatial data types do not exist in spatial data. For example, the lack of ordering in spatial data poses great challenges on spatial databases [66]. Furthermore, the relationships in spatial data types are often implicit, which makes it hard to detect, represent, and use them. For example, the relationships between two points, such as the distance and direction, are implicitly in their coordinates.

Another challenge comes from the properties of spatial data, such as spatial autocorrelation and spatial heterogeneity. Tobler's first law of geography, "everything is related to everything else, but near things are more related than distant things" [78], describes the spatial dependence that ubiquitously exists in the phenomena on earth. For example, people living in the same neighborhood tend to share similar characteristics, income, and education level. In spatial statistics, spatial dependence is called the spatial autocorrelation effect. Ignoring autocorrelation and assuming an identical and independent distribution (i.i.d.) of data when analyzing spatial data may produce hypotheses or models that are inaccurate [68]. For example, applying non-spatial machine learning methods (e.g., random forest) on land cover segmentation using remote sensing imagery may result in salt and pepper noise [42]. Spatial dependence exists not only at close locations but also distant locations. One example of long-range spatial dependence is El Nino and La Nina effects in the climate system. Spatial heterogeneity refers to the fact that spatial data do not follow an identical distribution throughout the entire earth. For example, the appearance information of European and Spiny Toads are visually similar, but they are located in different geographical regions and belong to different categories [52], which makes "one-size-fits-all" models using appearance information only hardly applicable.

Furthermore, while non-spatial data mining and machine learning techniques need discrete input data, for example, transactions in association rule mining, spatial datasets are embedded in continuous space, which makes the non-spatial techniques not applicable. The discretization of space may introduce problems such as the modifiable area unit problem (MAUP) or the multi-scale effect, since the results of spatial analysis depend on the choice of discretization methods and spatial scale [85]. Figure 1a shows three input spatial feature types A, B, and C and the nearby relationship in between. Depending on the choice of discretization methods as shown in Fig. 1b and c, the correlation coefficients of the pairs (A,B) and (B,C) are -1 and 1, respectively. Gerrymandering, which is a practice intended to establish a political advantage for a particular party or group by manipulating district boundaries, is a form of the MAUP and is attracting growing attention in recent years [19, 74].

There exists little literature on spatial data science. Xie et al. take a transdisciplinary perspective to present the foundations of spatial data science in mathematics, statistics, and computer science [84]. Most other related work focuses on spatial/spatiotemporal data mining, which is a phase of spatial data science. Atluri et al. [8] give the most recent review of spatiotemporal data mining, which is closely related to spatial data science, by surveying typical spatiotemporal data types and their properties, and by reviewing the approaches for commonly studied spatiotemporal data mining problems. In [68], Shekhar et al. review the

common spatiotemporal pattern families and their statistics background from a database-centric perspective. On approaches for mining purely spatial data there is a considerable literature. For example, Shekhar et al. survey the computational approaches to detect spatial outliers, co-location patterns, spatial classification and regression, spatial clustering, and spatial hotspots in [67]. There is also extensive literature on spatial or spatiotemporal statistics [18, 31], and data mining on specific types of spatial or spatiotemporal data (e.g., trajectory [95] and remote sensing imagery [47]). In this section, we discuss the life cycle of spatial data science comprehensively.

This chapter focuses on the spatial aspect of spatial data science but leaves the temporal aspect out of the scope, since it means to provide an introduction on spatial data science to the broad audiences, but there is no commonly accepted taxonomy for the temporal aspect of spatial data science currently. Much of the temporal aspect of spatial data science still needs to be explored further.

The rest of this chapter is organized as follows: Sect. 2 lists the spatial data types and the major sources of spatial data. Section 3 introduces spatial database for the storage and management of spatial data. Section 4 discusses the common pattern families of spatial data mining. Section 5 presents the methods of results validation, which is followed by the conclusion in Sect. 6.

2 Spatial Data

2.1 Types of Spatial Data

The major difference between spatial data and non-spatial data is that in addition to non-spatial information, spatial data contain spatial attributes, such as the location and the shape of a spatial object on earth. There are three common representations for spatial attributes, namely object models, field models, and spatial network models [66, 61].

An object model represents discrete objects as three basic data types, point(s), line(s), and polygon(s). These types were originally introduced by cartographers to represent real-world object in maps, so their use is decided by the level of detail to be represented. A point represents the geographic coordinates of a location, which could be a latitude and longitude in a geographic coordinate system or a x and y in a projected coordinate system. A point is often used to mark the location of an object, such as a city center (Fig. 2a). If the length of an object is of concern, for example, when showing a river in a small-scale map (Fig. 2b), the object will be represented by lines, which connect a sequence of points. A polygon is used when both the location and shape of an object need to be represented, such as a state on a map of the United States (Fig. 2c). A polygon is represented by lines connecting a sequence of points and close it. Now that the data types are more complex, the relationships between spatial data are also more complex and are frequently implicit in the spatial attributes. These relationships fall in four groups (as shown in Table 2).

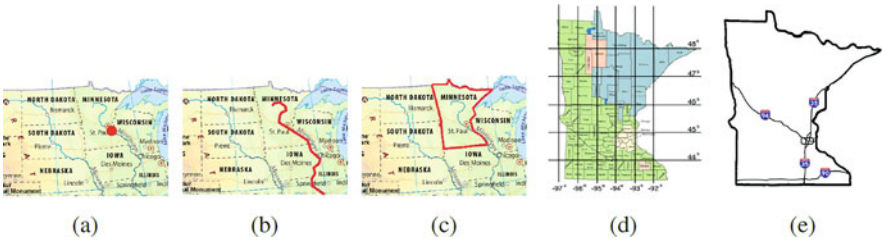


Fig. 2 Examples of the representations of spatial data. (**a**) St. Paul (Point). (**b**) Mississippi River (Line). (**c**) Minnesota State (Polygon). (**d**) Minnesota State (Field). (**e**) Interstate in Minnesota (Network)

Table 2 Relationships between spatial data

	Relationship
Set/Topological	union, intersection, meet, within, ...
Metric	distance, area, perimeter,...
Directional	above, left, eastern,...
Other	visibility,...

A field model is a mapping between continuous coordinates and a certain attribute. The value of the attribute may be categorical (e.g., land cover / land use) or numerical (e.g., rainfall amount). To be represented in computers, continuous space is often discretized using space partitioning. The partitioning is usually a regular grid with cells of the same size (e.g., a partitioning based on the coordinate system in Fig. 2d), but it does not have to be. Remote sensing imagery is a typical type of spatial data in the field model.

A spatial network model is composed of a set of points and the lines connecting them, which are called nodes and edges, respectively (Fig. 2e). Its major difference from the object model is its capability of representing the connectivity of the points and lines. Unlike a graph in discrete mathematics, the nodes and edges convey rich information. For example, a spatial network can represent a road system in an area, whose nodes may have information about turn restrictions, traffic control rules, etc., and edges may record the number of lanes, the lane width, and so forth.

2.2 Major Sources of Spatial Data

The most traditional way of acquiring spatial data is surveying, whose origin dates back to the beginning of recorded history [90]. The function of surveying includes determining and measuring spatial objects, assembling information related to the objects, and using the information for planning [39]. Because of its importance, surveying is mostly conducted by the government, and government agencies are

major suppliers of spatial data. The digitization of old paper maps provides historical spatial information.

A second important source of spatial data is remote sensing, both active and passive [14, 79]. Active remote sensing instruments, such as LIDAR and RADAR, have their own source of radiation, while passive instruments generally rely on sunlight. The imagery obtained from active and passive instruments is generated in fundamentally different ways, so they two are complementary and offer different perspectives of the Earth [43]. The advantages of passive remote sensing include that it is relatively easy to be interpreted and that it is of high spectral resolution. By contrast, LIDAR data, often in the form of point clouds, is widely used in reconstructing the surface of spatial objects or the earth [83]. One of the major advantages of RADAR is its penetration capacity, which enables it to work in various weather conditions and to detect targets under the Earth's surface [72].

The popularity of devices equipped with GPS chips (e.g., cellphones, watches) has boosted spatial data generation through crowdsourcing. For example, OpenStreetMap is a crowdsourced map built by volunteers around the world, which provides base maps to a large number of spatial data science research (e.g., [91]). Applications that users can use to check in with their locations such as Yelp and Twitter provide spatial data with rich non-spatial attributes and attract considerable research interest [16].

3 Spatial Database

Spatial database management systems (SDBMSs) are software modules that work with an underlying database management system (DBMS) to define, create, query, update, and manage a spatial database. These systems provide persistence across failures, concurrency control, and scalability to search queries on datasets that do not fit inside main memories of computers [66]. The SDBMS is an essential component of spatial data storage and management.

Compared with non-spatial DBMSs, SDBMSs can conduct operations that are specific for spatial data. Examples of the operations in the OGC (Open Geospatial Consortium) Standard for SQL [60] are listed in Table 3 in three families. The basic functions include operations on a single object. For example, the envelop operation yields the minimum bounding rectangle of an object. The topological/set operators examine the topological/set relationships between objects. The spatial analysis operations, on the other hand, contain commonly used spatial analysis such as generating a convex hull of a set of objects. Such operations allow SDBMSs to handle spatial queries that are beyond the capability of the non-spatial DBMS, for example, listing the fast food restaurants within one kilometer of a movie theater.

In order to facilitate spatial queries and the operations supporting them, the physical model of the SDBMS needs to be adjusted for spatial data, since non-spatial DBMSs cannot handle multi-dimensional data efficiently. In the non-spatial DBMS, sorting and hashing are used in the file structure and index to improve the

Table 3 Sample operations in OGC Standard for SQL

Family	Operation
Basic functions	SpatialReference, Envelop, Export, IsEmpty, IsSimple, Boundary
Topological/Set operators	Equal, Disjoint, Intersect, Touch, Cross, Within, Contains, Overlap, Union, Difference, SymmDiff
Spatial analysis	Distance, Buffer, ConvexHull

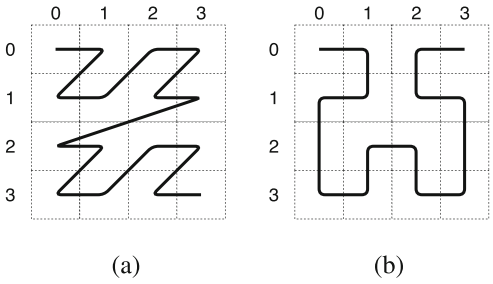


Fig. 3 Examples of space filling curves. (**a**) Z-order curve. (**b**) Hilbert's curve

computational performance of its basic building blocks such as point query, range query, join, insert, etc. [27]. However, the order of multi-dimensional data is not well defined, for example, there is no consensus on whether a location is greater or less than another to the southwest of it, and the hashing results may lose the spatial relationship between objects, which is important for spatial operations such as Intersect, Within, etc. Therefore, the file structure and index of the SDBMS need to be handled specifically.

The commonly used techniques in the file structure and index of the SDBMS include geo-hashing, quadtrees, and hierarchical collection of rectangles. The idea of geo-hashing is to use a space filling curve to connect all the cells of a space partition, usually a grid with cells of the same size, so that the cells can be sorted in the order in which the curve visits them [66]. The Z-order curve [55] and Hilbert's curve [34] are popular strategies for space filling curve. Both strategies map multi-dimensional data to one dimension while partially preserving locality of the data. Figure 3 shows the Z-order curve and Hilbert's curve for a 4×4 space partition of a given study area. Let the cell (i, j) be the cell in row i and column j. The Z-order curve preserves the proximity relationship between the cells $(1, 0)$ and $(1, 1)$ but breaks the relationship between $(1, 0)$ and $(2, 0)$. And there is a jump between $(1, 3)$ and $(2, 1)$, which are neighbors along the curve, but away from each other in reality. By contrast, there is no jump in Hilbert's curve, that is, the cells that are adjacent along the curve are also adjacent in reality. However, there is still a loss of proximity relationships, for example, that between the cells $(0, 1)$ and $(0, 2)$.

A quadtree is a tree data structure where each internal node has exactly four children [30]. A quadtree decomposes space into adaptable cells, each of which has a maximum capacity. When the maximum capacity of a cell is reached, the cell

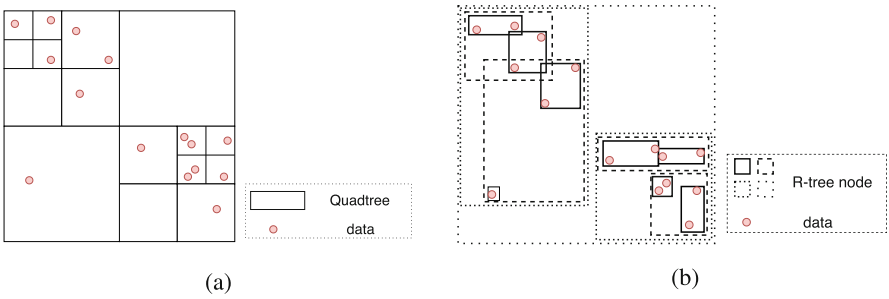


Fig. 4 Examples of the file structure and index in Spatial Database. (**a**) Quadtree. (**b**) R-tree

splits. Figure 4a shows a quadtree in which the maximum capacity of the leaf nodes is 2 data points. The ability of the quadtree to determine whether a cell should be split based on the density of the data inside it saves storage space, compared to having grid cells all the same size.

The R-tree structure is an example of a technique that uses a hierarchical collection of rectangles, which is a generalization of the B-tree to spatial data [33]. The R-tree is a balanced tree, and each node represents a rectangle. The rectangle of a child node is located within that of its parent, and the rectangles may overlap. If each node in an R-tree may have at most two children, a possible R-tree structure for the same point dataset in Fig. 4a is shown in Fig. 4b. There are a large number of R-tree variants. R+ trees only allow overlap between rectangles at the leaf level and duplicate the data so that a spatial object may exist in multiple leaf nodes [65]. R* trees allow data to be reinserted so that the overlap between rectangles is minimized [10]. Both of these variants have higher construction cost but lower query cost than the original R-tree.

In recent years, research on spatial database has mainly focused on improving computational performance using high-performance-computing platforms. Because of its capability of handling big data in commodity computer clusters, Apache Hadoop has been used by researchers and savvy tool users in various ways [71]. Some use Hadoop as a black box for operations on data, such as GIS tools for Hadoop, a package composed of both programming libraries and an add-on toolbox of ArcGIS desktop [28]; and Hadoop-GIS, a scalable spatial data warehousing system [5]. Spatial Hadoop, by contrast, adds native support for spatial data by supporting a set of spatial index structures and developing spatial functions that interact directly with Hadoop base code [92]. Impala, a distributed SQL query engine for Hadoop, has also been extended for spatial data [26]. Since the Hadoop-based tools are inefficient when handling interactive operations, GeoSpark [94] is developed to take full advantage of Apache Spark's core in-memory data abstraction, resilient distributed datasets.

4 Spatial Data Mining

Spatial data mining is the process of quantifying and discovering previously unknown, but potentially useful patterns from spatial data [68]. It mainly focuses on five pattern families, namely spatial hotspots, co-locations, spatial outliers, spatial prediction, and spatial change.

4.1 Spatial Hotspots

Given a set of spatial points, each of which is a spatial object or an event in a domain, a spatial hotspot is an area that has a higher probability density of the points compared to other parts in the study area [86]. Hotspot detection has important applications in many domains including public health [45], transportation [75], crime analysis [22], etc. John Snow's finding of the 1854 London cholera hotspot on a map was a major milestone in the development of the germ theory and saved numerous lives [86]. The main challenge of spatial hotspot detection arises from the uncertainty of the number, location, size, and shape of hotspots.

The research on spatial hotspots focuses on two perspectives of the problem, namely the interest measure that describes the probability density of the spatial points in an area, and the methods to enumerate the continuous space. Neill and Moore [59] named three properties of the interest measures for spatial hotspots: (1) For a fixed number of event in total, the interest measure for an area increases monotonically with the number of events in the area; (2) For a fixed number of events, the interest measure for an area decreases monotonically with the size of the population in the area; and (3) For a fixed proportion of events in a population, the interest measure for an area increases monotonically with the population in the area. Ever since the log likelihood radio (LLR) was introduced in [45], it has become one of the most popular interest measures for spatial hotspots. The LLR of an area describes the difference between the maximum likelihood estimate (MLE) of the probability density of the spatial points inside the area being higher than that outside the area, and the MLE of the probability density being the same throughout the study area. The greater the difference, the area is more likely to be a hotspot. Another popular interest measure that is simpler to compute than LLR is the density ratio [75, 70]. The density of an area is defined as the number of events normalized with the population in the area, so the density ratio of an area depicts the difference between the density inside and outside the area. Given the importance of the interest measure, which has direct impact on the solution quality of hotspot detection, research about interest measures is still active and multiple efforts have been made to identify the theoretical limitations of the current measures (e.g., likelihood ratio) and address them. Examples of more recent interest measures include nondeterministic normalization index [86] and expectation-based likelihood ratio [58]. In all the previous examples of interest measure, the geometric area (or

size of the population) of a hotspot is required in the calculation. However, in non-spatial clustering techniques such as DBSCAN the area or the size of the population is often unknown or hard to calculate, since the outputs are often clustered point-sets rather than a well-defined spatial region. Thus, new interest measure is introduced for clustering methods (e.g., cluster size [87]). In spatial statistics, local indicators of spatial association, including local Moran's I, Ripley's K, Geary's C, Ord Gi, and Gi* functions [6], are also used to indicate hotspots.

Research is also conducted on the methods of identifying hotspots in continuous space. Most of the spatial hotspot detection studies use spatial scan statistic methods. The spatial scan statistic extends the original scan statistic for a one-dimensional point process to allow for multi-dimensional space [45]. In order to enumerate through continuous space, researchers have come up with different definitions of the shape and the location of the potential hotspots. In Euclidean space, the definitions include circles with one spatial point at the center and another one on the perimeter [45], rectangles in a grid [59], ellipses [76], and ring-shape area [24]. In spatial networks, the examples are linear intersecting paths [70], constrained minimum spanning trees [17], isodistance spanning trees [77], isodistance spanning trees with a hole [23], and shortest paths [75]. Given a certain definition of the potential hotspots, the focus of these papers is on improving the computational performance of the proposed algorithms. Since the common interest measures, e.g., LLR and density ratio, do not obey the monotonicity property, meaning that there is no ordering between the interest measures in an area and its sub-area, or vice versa, some researchers have proposed upper and lower bounds of the interest measures, which do obey the monotonicity property [59, 75, 24]. Based on the proposed lower and upper bounds, their algorithms apply strategies such as divide and conquer [59], and filter and refine [75, 24]. Clustering based methods are also used to find candidate areas for further evaluation, including global partitioning based methods (e.g., the K-Means and the K-Medoids [44]), hierarchical clustering methods (e.g., robust clustering using links (ROCK) [32]), and density-based methods (e.g., DBSCAN [29, 87]).

4.2 Co-locations

A co-location pattern is a set of spatial object types whose instances are frequently located in geographic proximity [36]. Real-world examples include symbiotic species, such as the Nile crocodile and Egyptian plover, and other biological dependencies (e.g., the dependence between different types of blackberry canes [9]). The challenges of mining co-location patterns are two-fold: first, there is no explicit partitioning in continuous space; and second, the number of candidate patterns is exponential.

In spatial statistics, the cross-K function is a generalization of Ripley's K function for a multivariate spatial point process, which depicts the spatial correlation of different types of spatial objects [21]. The cross-K function of two spatial object types i and j is defined as

$$K_{ij}(h) = \frac{E(\#\text{ type } j \text{ instances within distance } h \text{ of a type } i \text{ instance})}{\lambda_j}, \quad (1)$$

where λ_j is the number of type j instances per unit area, and $E(\cdot)$ is a function to calculate the expectation. When there are n spatial object types, there would be n^2 cross-K functions for them. A higher cross-K function value indicates more instances of type i in the proximity of each type j instance on average, that is, i and j are more likely to be co-location pattern.

It is hard to generalize the cross-K function to a set of more than two spatial object types, and the computational cost of the cross-K function is high. An alternative interest measure for co-location pattern detection is the participation index (PI) [36]. The PI is defined based on the definition of the participation ratio (PR). If every pair of objects in a set of spatial objects are neighbors of each other, we say the set of objects form a clique. Given a subset S of the spatial object types, the PR of type $t \in S$ is defined as the proportion of instances of t that form cliques with the instances of all other types in S. The PI of a subset S is the minimum PR of every type $t \in S$. The greater the value of the PI of a co-location pattern, the more likely that instances of the types in the pattern are located in geographic proximity. The participation index of two spatial object types A, B is an upper bound of $\frac{K_{AB}(h)}{W}$, where $K_{AB}(h)$ is the estimation of the cross-K function of the set A, B for a proximity neighborhood defined by distance $\leq h$, and W is the total area of the study area [36]. In addition, since the participation index of a set is always less than or equal to that of its subset. The Apriori algorithm for association rule detection [4] can be applied to avoid the redundant computation for finding potential patterns.

Researchers have defined aspects of the co-location detection problem in different ways. First, the definition of co-location pattern is extended. Research in this direction include generalizing the data type of the spatial object from points to points, lines, and polygons [89]; loosening the requirements of co-location patterns by removing the support threshold [37]; mining the patterns with rare events [35]; making the neighborhood constraint dynamic [64]; and defining a neighborhood using a k-nearest neighbor graph [63]. Second, the computational performance of the algorithms for mining the pattern is improved [93]. Third, the spatial heterogeneity is taken into consideration, and mining co-location pattern in local regions is studied. Depending on how the local regions are generated, these include studies using data-unaware heuristics such as quadtree [15] and regular grid [25, 81], and others using data-aware heuristics such as neighbor graph [54], Delaunay triangulation [20, 12], and minimum orthogonal bounding rectangles [49].

4.3 Spatial Outliers

Spatial outliers are spatial data records (i.e., points, lines, polygons) that have different non-spatial attribute values from their spatial neighborhood [3]. For example,

according to the age of houses, a new house in an old community can be considered as a spatial outlier. Spatial outlier detection is vital for applications that need to find unusual or suspicious activity or objects compared to their neighborhoods. Such applications include anomalous traffic monitoring in transportation engineering, fraud detection and prediction in credit card transactions and suspicious object and behavior detection in criminology.

There are two categories of methods for detecting spatial outliers, graphical and quantitative. Graphical tests detect outliers by considering visualized patterns from data. Examples include the variogram cloud plot and the Moran scatter plot. In the variogram cloud plot, the x axis is the pairwise geographic distance, and the y axis is the square root of the absolute difference of attribute values. Records that may contain spatial outliers can be found by identifying the points that represent pairs with short geographic distance and great difference of attribute values [62]. In a Moran scatter plot, the x axis is the Z-score of the attribute value, which is the deviation of the value from the mean, and the y axis is the weighted average Z-score of the neighbors' attribute values [7]. The spatial outliers locate away from the line $y = x$ in the plot. The second type of methods to detect outliers are the quantitative methods, which calculate the difference between the attribute values of inspected points and their spatial neighbors. When the difference is larger than a predefined threshold, an outlier is detected. For example, a spatial Z-score, which is a variant of Z-score, is the deviation of the attribute values of inspected points from the mean of their spatial neighbors.

4.4 Spatial Prediction

Spatial prediction is the process of learning a model and predicting the target variable at a specific location using the explanatory variables at the location, training samples at other locations, and the relationships between locations [69]. Its applications are found in various domains such as environmental studies, land management, etc. The main challenges of spatial prediction are caused by the autocorrelation and heterogeneity of spatial data [68]. Table 4 lists examples of non-spatial prediction models and their variants for spatial prediction problems, which consider spatial autocorrelation.

Table 4 Spatial autocorrelation in prediction models

Non-spatial	Spatial
$y = X\beta + \epsilon$	$y = \rho Wy + X\beta + \epsilon$
$Pr(c_i\|X) = \frac{Pr(X\|c_i)Pr(c_i)}{Pr(X)}$	$Pr(c_i\|X, C_N) = \frac{Pr(X,C_N\|c_i)Pr(c_i)}{Pr(X,C_N)}$
Neural networks	Convolutional neural networks
Decision trees	Spatial decision trees

Non-spatial prediction approaches assume that samples are from an identical and independent distribution (i.i.d.), which does not hold for spatial data. Using non-spatial tools for spatial prediction task may result in a large number of prediction errors, such as salt and pepper noise [42], and errors caused by spatial heterogeneity [41].

Spatial properties are explicitly modeled in spatial regression models to handle spatial autocorrelation and heterogeneity. The spatial auto-regressive (SAR) model belongs to this model family, in which a neighborhood relationship between the target variables is added as an input [73]. The SAR model is defined as follows:

$$y = \rho W y + X\beta + \epsilon, \tag{2}$$

where W is an adjacency matrix, and Wy models the effect of the neighborhood in addition to the effect of the explanatory variables X on the target variable y, and ρ and β are the parameters that should be learned. Notice that linear regression, which does follow the i.i.d assumption, is a special case of the SAR model when ρ is zero. So, the SAR model is a more general model than the linear regression model. Geographically weighted regression (GWR) is another regression model that considers spatial autocorrelation [11]. GWR does not do a regression on all data samples. It uses a focal window centered at the current location and fits a regression model only to the training samples in the window. Samples that are closer to the current location in the window will get more weight. Similarly, there are approaches that incorporate spatial context into classification problems, such as Markov random fields (MRF) [69]. In recent years, spatial prediction using deep neural network attracts growing attention. For example, Lin et al. introduce a deep neural network model for location dependent time series prediction, which explicitly incorporates spatial autocorrelation into neural networks to deal with sparse and unevenly distributed observations [51].

4.5 Spatial Change

Given a definition of change, spatial change discovery is the process of identifying the location of such changes from the dataset [96]. While the definition of spatial change patterns varies with the applications, most change patterns fall into four groups: changes in the statistic parameters modeling the phenomena, changes in the actual observation values, changes in the models that fit the data, and changes in derived attributes.

Detecting patterns of spatial changes between snapshots is an important problem in spatial changes detection. In remote sensing, detecting changes between satellite images can help identify land cover change due to human activity, natural disasters, or climate change, so the change footprint detection using two images of the same area taken at different time is extensively studied. The change footprints include local footprints (e.g., pixels in images), focal footprints (e.g., groups of pixels,

called blocks, in images [2]), and object-based areas [38]. In addition, given a spatiotemporal path, identifying its sub-paths where intensive changes occur is also studied [97]. The path can be in both temporal dimension, for example, the amount of monthly rainfall at a location in 10 years, and spatial dimension, for example, the amount of vegetation cover along a meridian on earth.

5 Result Validation

In the last few years, the fairness, accountability, transparency, and ethics in data science, AI, and machine learning are attracting growing more attention. In many important societal applications, false positives (i.e., spurious patterns) are often associated with high economic and social cost. A false alarm about a disease outbreak may cause huge wastes of resources to "control" it and raise anxiety in the population. Therefore, results of spatial data mining have to be validated further. Common methods for result validation include integrating prior domain knowledge, acquiring additional data, and statistical significance testing. In the rest of this section, we introduce the use of significance testing in spatial pattern mining, covering the key concepts, formulations, and computational techniques.

5.1 Key Concepts

Generation process of data The statistical process that governs how the data is generated. In the case of hotspot detection, the data is a spatial distribution of points, so the generation process (a.k.a. point process) specifies the probability density of each point being located at each spatial location. The distribution of probability densities determines whether a true hotspot exists or not.

Null and alternative hypotheses The null hypothesis states that there is no significant difference between two populations, where, for spatial pattern mining, one population represents the subset of data associated with the detected patterns and the other represents the rest of the data. The alternative hypothesis states that the two populations are significantly different (e.g., generated by different processes).

Test statistic A random variable is used in hypothesis testing. Its value is computed from the data samples representing a population. The test statistic values of two populations (i.e., data belonging or not belonging to a detected pattern) are used to determine whether to reject the null hypothesis.

5.2 Formulations and Computational Techniques

We use spatial hotspot detection to illustrate a concrete formulation of statistical significance and related computational techniques. We also briefly discuss recent developments in significance testing for other patterns (i.e., co-location, segregation, and change footprints).

5.2.1 Spatial Hotspots

Spatial hotspot detection aims to find sub-regions of a study area that have a higher probability density of generating points (e.g., crime or disease cases) than the rest of the area. Based on this definition, the null hypothesis states that there is no significant difference between the probability densities of point-generation in different sub-regions. In other words, the point-generation process is a homogeneous process across the entire study area. If a point distribution follows the null hypothesis, then any high-density regions are just chance patterns formed randomly in the generation process.

Denote $f_t()$ as the choice of a test statistic, and h as a detected hotspot in observed (input) data with a test statistic value of $f_t(h)$. Further denote $f_t(h')$ as the test statistic value of a sub-region in a point distribution generated by the null hypothesis (e.g., a homogeneous point process). The p-value of h is then the probability that there exists a $f_t(h')$ that is higher than $f_t(h)$.

Since the p-value often cannot be calculated in closed-form using existing statistical models [45, 59, 86, 58, 87], we compute it using Monte Carlo simulation. The Monte Carlo simulation has M trials. In each trial, a random point distribution is generated using the null hypothesis and the same hotspot detection algorithm used for the observed data is applied to find the chance pattern that has the highest test statistic value. Upon completion of all M trials, we have M such highest values. Denote the desired significance level as α. If $f_t(h)$ is among the highest αM values, then we reject the null hypothesis and conclude that the pattern is significant; otherwise, a chance pattern. Monte Carlo simulation is computationally challenging because it requires a large number of repetitive runs of the detection algorithm (e.g., $M = 1000, 10{,}000$). Various techniques have been developed to accelerate Monte Carlo simulation. Many of these methods propose efficiently calculable upper and lower bounds of test statistic values [59, 86, 58, 87], which can be used to prune subsets during candidate enumeration or reduce the number of runs of the exact detection algorithm (e.g., exact DBSCAN). An early termination technique is also used to halt the simulation if a conclusion can already be drawn from existing trials (e.g., already more than αM trials with a larger test statistic value).

5.2.2 Other Patterns

The importance of statistical robustness has also been recognized for other patterns, such as co-location, segregation, and change detection. For patterns of co-location and segregation (i.e., points belonging to different features repel each other rather than co-locate together), the common test statistic adopted is the participation index [36]. The only difference is that significant patterns of segregation should have a very low participation index rather than a high index. Multiple formulations of the null hypothesis were explored in [9], including homogeneous point processes and clustered point processes. The same Monte Carlo framework is used to compute the p-values of detected co-location and segregation patterns. In change detection, a major issue is that many detected change footprints are very short in length and thus are more likely to be caused by random local fluctuations or measurement errors rather than persistent trend of changes. A recent study [88] proposed a significance formulation using pattern size as the test statistic (e.g., the length of a sub-path of change in the one-dimensional case). The null hypothesis is that the changes along an input path are caused by a random distribution of values, and there is no sub-path that has a higher probability of having changes. This formulation can filter out local fluctuations and keep meaningful persistent trends. This work also proposed a dynamic search-and-prune algorithm to build connections between different subsets of paths for pruning, which speeds up the Monte Carlo simulation. Case studies showed that the incorporation of significance can separate meaningful ecotones from random noise using African NDVI data.

6 Research Trend

In the last decade, the utilization of both spatial and temporal information in observation data is growing popular. Extensive research has been conducted on an emerging field of spatiotemporal data mining. The methods are introduced to detect spatiotemporal patterns, such as spatiotemporal outliers, association and tele-coupling, prediction, forecasting, partitioning and summarization, etc.

Some cutting edge research is also conducted on spatial data mining in spatial networks. For example, several spatial network statistical methods have been developed, e.g., network K function and network spatial autocorrelation. Another example is the linear hotspot discovery problem that aims to find paths in a spatial network as hotspots. Another growing research area is spatial prediction in the spatial network using GPS trajectories and on-board diagnostics (OBD) data from vehicles is also widely studied. For example, Li et al. [50, 48] propose an energy-efficient path selection algorithm that uses a path-centric energy consumption model that predicts the expected energy consumption of traveling based on the historical OBD datasets.

7 Conclusion

The growing availability of spatial data is spurring the emergence and development of spatial data science, a multi-disciplinary field that applies scientific methods to retrieve previously unknown, but potentially useful and non-trivial knowledge and insights from spatial data. It is important for societal applications in various domains such as public health, public safety, agriculture, environmental science, climate studies, etc. Computerized methods are needed to store and manage spatial data, to retrieve knowledge from the data, and to validate the results. The challenges of spatial data science arise from its interdisciplinary nature and the unique properties of spatial data. In this section, we provided an introduction to the spatial aspect of spatial data science along its life cycle: data acquisition, data storage, data mining, and result validation and then listed two research trends including the integration of temporal information as well as generalizing spatial data science methods into spatial network space.

References

1. GIS At DOT (2017). https://www.transportation.gov/gis
2. Aach, T., Kaup, A., Mester, R.: Statistical model-based change detection in moving video. Signal processing **31**(2), 165–180 (1993)
3. Aggarwal, C.C.: Outlier analysis. In: Data mining, pp. 237–263. Springer (2015)
4. Agrawal, R., Srikant, R., others: Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB, vol. 1215, pp. 487–499 (1994)
5. Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X., Saltz, J.: Hadoop GIS: a high performance spatial data warehousing system over MapReduce. Proceedings of the VLDB Endowment **6**(11), 1009–1020 (2013)
6. Anselin, L.: Local indicators of spatial association—LISA. Geographical analysis **27**(2), 93–115 (1995)
7. Anselin, L.: The Moran scatterplot as an ESDA tool to assess local instability in spatial association. In: Spatial Analytical, pp. 111–126. Routledge (2019)
8. Atluri, G., Karpatne, A., Kumar, V.: Spatio-Temporal Data Mining: A Survey of Problems and Methods. ACM Comput. Surv. **51**(4), 83:1–83:41 (2018). https://doi.org/10.1145/3161602
9. Barua, S., Sander, J.: Mining statistically significant co-location and segregation patterns. IEEE Transactions on Knowledge and Data Engineering **26**(5), 1185–1199 (2013)
10. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The r*-tree: an efficient and robust access method for points and rectangles. In: ACM SIGMOD Record, vol. 19, pp. 322–331. ACM (1990)
11. Brunsdon, C., Fotheringham, S., Charlton, M.: Geographically weighted regression. Journal of the Royal Statistical Society: Series D (The Statistician) **47**(3), 431–443 (1998)
12. Cai, J., Liu, Q., Deng, M., Tang, J., He, Z.: Adaptive detection of statistically significant regional spatial co-location patterns. Computers, Environment and Urban Systems **68**, 53–63 (2018). https://doi.org/10.1016/j.compenvurbsys.2017.10.003
13. Caldwell, P.M., Bretherton, C.S., Zelinka, M.D., Klein, S.A., Santer, B.D., Sanderson, B.M.: Statistical significance of climate sensitivity predictors obtained by data mining. Geophysical Research Letters **41**(5), 1803–1808 (2014). https://doi.org/10.1002/2014GL059205

14. Campbell, J.B., Wynne, R.H.: Introduction to Remote Sensing, Fifth Edition, 5th edition edn. The Guilford Press, New York (2011)
15. Celik, M., Kang, J.M., Shekhar, S.: Zonal co-location pattern discovery with dynamic parameters. In: Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on, pp. 433–438. IEEE (2007)
16. Cheng, Z., Caverlee, J., Lee, K.: You Are Where You Tweet: A Content-based Approach to Geo-locating Twitter Users. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, pp. 759–768. ACM, New York, NY, USA (2010). https://doi.org/10.1145/1871437.1871535. Event-place: Toronto, ON, Canada
17. Costa, M.A., Assunção, R.M., Kulldorff, M.: Constrained spanning tree algorithms for irregularly-shaped spatial clustering. Computational Statistics & Data Analysis **56**(6), 1771–1783 (2012). https://doi.org/10.1016/j.csda.2011.11.001
18. Cressie, N.: Statistics for Spatial Data. John Wiley & Sons (2015)
19. Daley, D.: GOP Racial Gerrymandering Mastermind Participated in Redistricting in More States Than Previously Known, Files Reveal (2019). https://theintercept.com/2019/09/23/gerrymandering-gop-west-virginia-florida-alabama/
20. Deng, M., Cai, J., Liu, Q., He, Z., Tang, J.: Multi-level method for discovery of regional co-location patterns. International Journal of Geographical Information Science **31**(9), 1846–1870 (2017). https://doi.org/10.1080/13658816.2017.1334890
21. Dixon, P.M.: Ripley's K Function. In: Encyclopedia of Environmetrics. John Wiley & Sons, Ltd (2006). https://doi.org/10.1002/9780470057339.var046
22. Eck, J., Chainey, S., Cameron, J., Wilson, R.: Mapping crime: Understanding hotspots (2005). http://discovery.ucl.ac.uk/11291/1/11291.pdf
23. Eftelioglu, E., Li, Y., Tang, X., Shekhar, S., Kang, J.M., Farah, C.: Mining Network Hotspots with Holes: A Summary of Results. In: Geographic Information Science, Lecture Notes in Computer Science, pp. 51–67. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45738-3_4
24. Eftelioglu, E., Shekhar, S., Kang, J.M., Farah, C.C.: Ring-Shaped Hotspot Detection. IEEE Transactions on Knowledge and Data Engineering **28**(12), 3367–3381 (2016). https://doi.org/10.1109/TKDE.2016.2607202
25. Eick, C.F., Parmar, R., Ding, W., Stepinski, T.F., Nicot, J.P.: Finding Regional Co-location Patterns for Sets of Continuous Variables in Spatial Datasets. In: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '08, pp. 30:1–30:10. ACM, New York, NY, USA (2008). https://doi.org/10.1145/1463434.1463472
26. Eldawy, A., Mokbel, M.F.: SpatialHadoop: A MapReduce framework for spatial data. In: 2015 IEEE 31st International Conference on Data Engineering, pp. 1352–1363 (2015). https://doi.org/10.1109/ICDE.2015.7113382
27. Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems, 7 edn. Pearson, Hoboken, NJ (2015)
28. ESRI: GIS Tools for Hadoop by Esri. http://esri.github.io/gis-tools-for-hadoop/
29. Ester, M., Kriegel, H.P., Sander, J., Xu, X., others: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, vol. 96, pp. 226–231 (1996)
30. Finkel, R.A., Bentley, J.L.: Quad trees a data structure for retrieval on composite keys. Acta informatica **4**(1), 1–9 (1974)
31. Gelfand, A.E., Diggle, P., Guttorp, P., Fuentes, M.: Handbook of Spatial Statistics. CRC Press (2010)
32. Guha, S., Rastogi, R., Shim, K.: Rock: A robust clustering algorithm for categorical attributes. Information systems **25**(5), 345–366 (2000)
33. Guttman, A.: R-trees: A dynamic index structure for spatial searching, vol. 14. ACM (1984)
34. Hilbert, D.: Über die stetige abbildung einer linie auf ein flächenstück. In: Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes, pp. 1–2. Springer (1935)

35. Huang, Y., Pei, J., Xiong, H.: Mining Co-Location Patterns with Rare Events from Spatial Data Sets. GeoInformatica **10**(3), 239–260 (2006). https://doi.org/10.1007/s10707-006-9827-8
36. Huang, Y., Shekhar, S., Xiong, H.: Discovering colocation patterns from spatial data sets: a general approach. IEEE Transactions on Knowledge and Data Engineering **16**(12), 1472–1485 (2004)
37. Huang, Y., Xiong, H., Shekhar, S., Pei, J.: Mining Confident Co-location Rules Without a Support Threshold. In: Proceedings of the 2003 ACM Symposium on Applied Computing, SAC '03, pp. 497–501. ACM, New York, NY, USA (2003). https://doi.org/10.1145/952532.952630
38. Im, J., Jensen, J., Tullis, J.: Object-based change detection using correlation image analysis and image segmentation. International Journal of Remote Sensing **29**(2), 399–423 (2008)
39. International Federation of Surveyors: FIG Definition of the Functions of the Surveyor (2004). http://www.fig.net/about/general/definition/index.asp
40. Jia, X., Willard, J., Karpatne, A., Read, J., Zwart, J., Steinbach, M., Kumar, V.: Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles. In: Proceedings of the 2019 SIAM International Conference on Data Mining, pp. 558–566. SIAM (2019)
41. Jiang, Z., Sainju, A.M., Li, Y., Shekhar, S., Knight, J.: Spatial ensemble learning for heterogeneous geographic data with class ambiguity. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(4), 43 (2019)
42. Jiang, Z., Shekhar, S., Zhou, X., Knight, J., Corcoran, J.: Focal-Test-Based Spatial Decision Tree Learning. IEEE Transactions on Knowledge and Data Engineering **27**(6), 1547–1559 (2015). https://doi.org/10.1109/TKDE.2014.2373383
43. Joshi, N., Baumann, M., Ehammer, A., Fensholt, R., Grogan, K., Hostert, P., Jepsen, M.R., Kuemmerle, T., Meyfroidt, P., Mitchard, E.T.A., Reiche, J., Ryan, C.M., Waske, B.: A Review of the Application of Optical and Radar Remote Sensing Data Fusion to Land Use Mapping and Monitoring. Remote Sensing **8**(1), 70 (2016). https://doi.org/10.3390/rs8010070
44. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis, vol. 344. John Wiley & Sons (2009)
45. Kulldorff, M.: A spatial scan statistic. Communications in Statistics—Theory and Methods **26**(6), 1481–1496 (1997). https://doi.org/10.1080/03610929708831995
46. Lens, M.C., Meltzer, R.: Is Crime Bad for Business? Crime and Commercial Property Values in New York City. Journal of Regional Science **56**(3), 442–470 (2016). https://doi.org/10.1111/jors.12254
47. Li, W., Du, Q.: A survey on representation-based classification and detection in hyperspectral remote sensing imagery. Pattern Recognition Letters **83**, 115–123 (2016)
48. Li, Y., Kotwal, P., Wang, P., Shekhar, S., Northrop, W.: Trajectory-aware Lowest-cost Path Selection: A Summary of Results. In: Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD '19, pp. 61–69. ACM, Vienna, Austria (2019). https://doi.org/10.1145/3340964.3340971
49. Li, Y., Shekhar, S.: Local Co-location Pattern Detection: A Summary of Results. In: S. Winter, A. Griffin, M. Sester (eds.) 10th International Conference on Geographic Information Science (GIScience 2018), *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 114, pp. 10:1–10:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). https://doi.org/10.4230/LIPIcs.GISCIENCE.2018.10
50. Li, Y., Shekhar, S., Wang, P., Northrop, W.: Physics-guided Energy-efficient Path Selection: A Summary of Results. In: Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL '18, pp. 99–108. ACM, Seattle, WA, USA (2018). https://doi.org/10.1145/3274895.3274933
51. Lin, Y., Chiang, Y.Y., Franklin, M., Eckel, S.P., Ambite, J.L.: Building autocorrelation-aware representations for fine-scale spatiotemporal prediction. In: 2020 IEEE International Conference on Data Mining (ICDM), pp. 352–361. IEEE (2020)

52. Mac Aodha, O., Cole, E., Perona, P.: Presence-only geographical priors for fine-grained image classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9596–9606 (2019)
53. Marcus, G., Davis, E.: Eight (No, Nine!) Problems With Big Data. The New York Times (2014). http://www.nytimes.com/2014/04/07/opinion/eight-no-nine-problems-with-big-data.html
54. Mohan, P., Shekhar, S., Shine, J.A., Rogers, J.P., Jiang, Z., Wayant, N.: A Neighborhood Graph Based Approach to Regional Co-location Pattern Discovery: A Summary of Results. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '11, pp. 122–132. ACM, New York, NY, USA (2011). https://doi.org/10.1145/2093973.2093991
55. Morton, G.M.: A computer oriented geodetic data base and a new technique in file sequencing (1966)
56. National Cancer Institute: GIS at the National Cancer Institute. https://gis.cancer.gov/gis-nci/gis_nci.html
57. National Geospatial-Intelligence Agency: About NGA. https://www.nga.mil/About/Pages/Default.aspx
58. Neill, D.B.: Expectation-based scan statistics for monitoring spatial time series data. International Journal of Forecasting **25**(3), 498–517 (2009)
59. Neill, D.B., Moore, A.W.: Rapid Detection of Significant Spatial Clusters. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04, pp. 256–265. ACM, New York, NY, USA (2004). https://doi.org/10.1145/1014052.1014082
60. Open Geospatial Consortium: OpenGIS Implementation Specification for Geographic information—Simple feature access—Part 2: SQL option. http://portal.opengeospatial.org/files/?artifact_id=25354
61. Open Geospatial Consortium: OGC Standards and Supporting Documents (2019). http://www.opengeospatial.org/standards/
62. Ploner, A.: The use of the variogram cloud in geostatistical modelling. Environmetrics: The official journal of the International Environmetrics Society **10**(4), 413–437 (1999)
63. Qian, F., Chiew, K., He, Q., Huang, H.: Mining regional co-location patterns with kNNG. Journal of Intelligent Information Systems **42**(3), 485–505 (2014). https://doi.org/10.1007/s10844-013-0280-5
64. Qian, F., He, Q., He, J.: Mining Spatial Co-location Patterns with Dynamic Neighborhood Constraint. In: Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, pp. 238–253. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04174-7_16
65. Sellis, T., Roussopoulos, N., Faloutsos, C.: The r+-tree: A dynamic index for multi-dimensional objects. Tech. rep. (1987)
66. Shekhar, S., Chawla, S.: Spatial Databases: A Tour, 1 edition edn. Prentice Hall, Upper Saddle River, N.J (2003)
67. Shekhar, S., Evans, M.R., Kang, J.M., Mohan, P.: Identifying patterns in spatial information: A survey of methods. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **1**(3), 193–214 (2011). https://doi.org/10.1002/widm.25
68. Shekhar, S., Jiang, Z., Ali, R., Eftelioglu, E., Tang, X., Gunturi, V., Zhou, X.: Spatiotemporal data mining: A computational perspective. ISPRS International Journal of Geo-Information **4**(4), 2306–2338 (2015)
69. Shekhar, S., Schrater, P.R., Vatsavai, R.R., Wu, W., Chawla, S.: Spatial contextual classification and prediction models for mining geospatial data. IEEE Transactions on Multimedia **4**(2), 174–188 (2002)
70. Shi, L., Janeja, V.P.: Anomalous Window Discovery for Linear Intersecting Paths. IEEE Transactions on Knowledge and Data Engineering **23**(12), 1857–1871 (2011). https://doi.org/10.1109/TKDE.2010.212

71. Shvachko, K., Kuang, H., Radia, S., Chansler, R., et al.: The Hadoop distributed file system. In: MSST, vol. 10, pp. 1–10 (2010)
72. Sinha, S., Jeganathan, C., Sharma, L.K., Nathawat, M.S.: A review of radar remote sensing for biomass estimation. International Journal of Environmental Science and Technology **12**(5), 1779–1792 (2015). https://doi.org/10.1007/s13762-015-0750-0
73. Srinivasan, S.: Spatial Regression Models. In: S. Shekhar, H. Xiong, X. Zhou (eds.) Encyclopedia of GIS, pp. 1–6. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-23519-6_1294-2
74. Stewart, A.J., Mosleh, M., Diakonova, M., Arechar, A.A., Rand, D.G., Plotkin, J.B.: Information gerrymandering and undemocratic decisions. Nature **573**(7772), 117–121 (2019). https://doi.org/10.1038/s41586-019-1507-6
75. Tang, X., Eftelioglu, E., Oliver, D., Shekhar, S.: Significant Linear Hotspot Discovery. IEEE Transactions on Big Data **3**(2), 140–153 (2017). https://doi.org/10.1109/TBDATA.2016.2631518
76. Tang, X., Eftelioglu, E., Shekhar, S.: Elliptical Hotspot Detection: A Summary of Results. In: Proceedings of the 4th International ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data, BigSpatial'15, pp. 15–24. ACM, New York, NY, USA (2015). https://doi.org/10.1145/2835185.2835192
77. Tang, X., Eftelioglu, E., Shekhar, S.: Detecting Isodistance Hotspots on Spatial Networks: A Summary of Results. In: M. Gertz, M. Renz, X. Zhou, E. Hoel, W.S. Ku, A. Voisard, C. Zhang, H. Chen, L. Tang, Y. Huang, C.T. Lu, S. Ravada (eds.) Advances in Spatial and Temporal Databases, Lecture Notes in Computer Science, pp. 281–299. Springer International Publishing (2017)
78. Tobler, W.R.: A Computer Movie Simulating Urban Growth in the Detroit Region. Economic Geography **46**(sup1), 234–240 (1970). https://doi.org/10.2307/143141
79. Toth, C., Jóźków, G.: Remote sensing platforms and sensors: A survey. ISPRS Journal of Photogrammetry and Remote Sensing **115**, 22–36 (2016). https://doi.org/10.1016/j.isprsjprs.2015.10.004
80. Walsh, B.: Google's Flu Project Shows the Failings of Big Data. https://time.com/23782/google-flu-trends-big-data-problems/
81. Wang, S., Huang, Y., Wang, X.S.: Regional Co-locations of Arbitrary Shapes. In: Advances in Spatial and Temporal Databases, pp. 19–37. Springer Berlin Heidelberg (2013). https://doi.org/10.1007/978-3-642-40235-7_2
82. Wong, C., Sorensen, P., Hollywood, J.S.: Evaluation of National Institute of Justice-Funded Geospatial Software Tools (2014). https://www.rand.org/pubs/research_reports/RR418.html
83. Wu, B., Yu, B., Wu, Q., Yao, S., Zhao, F., Mao, W., Wu, J.: A Graph-Based Approach for 3D Building Model Reconstruction from Airborne LiDAR Point Clouds. Remote Sensing **9**(1), 92 (2017). https://doi.org/10.3390/rs9010092
84. Xie, Y., Eftelioglu, E., Ali, R.Y., Tang, X., Li, Y., Doshi, R., Shekhar, S.: Transdisciplinary Foundations of Geospatial Data Science. ISPRS International Journal of Geo-Information **6**(12), 395 (2017)
85. Xie, Y., Gupta, J., Li, Y., Shekhar, S.: Transforming smart cities with spatial computing. In: 2018 IEEE International Smart Cities Conference (ISC2), pp. 1–9. IEEE (2018)
86. Xie, Y., Shekhar, S.: A Nondeterministic Normalization based Scan Statistic (NN-scan) towards Robust Hotspot Detection: A Summary of Results. In: Proceedings of the 2019 SIAM International Conference on Data Mining, Proceedings, pp. 82–90. Society for Industrial and Applied Mathematics (2019). https://doi.org/10.1137/1.9781611975673.10
87. Xie, Y., Shekhar, S.: Significant DBSCAN Towards Statistically Robust Clustering. In: Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD '19, pp. 31–40. ACM, New York, NY, USA (2019). https://doi.org/10.1145/3340964.3340968. Event-place: Vienna, Austria
88. Xie, Y., Zhou, X., Shekhar, S.: Discovering interesting sub-paths with statistical significance from spatio-temporal datasets. ACM Transactions on Intelligent Systems and Technology (2019)

89. Xiong, H., Shekhar, S., Huang, Y., Kumar, V., Ma, X., Yoc, J.: A Framework for Discovering Co-location Patterns in Data Sets with Extended Spatial Objects. In: Proceedings of the 2004 SIAM International Conference on Data Mining, Proceedings, pp. 78–89. Society for Industrial and Applied Mathematics (2004)
90. Yan, H.S., Ceccarelli, M. (eds.): International Symposium on History of Machines and Mechanisms: Proceedings of HMM 2008. History of Mechanism and Machine Science. Springer Netherlands (2009)
91. Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., Aberer, K.: Semantic Trajectories: Mobility Data Computation and Annotation. ACM Trans. Intell. Syst. Technol. **4**(3), 49:1–49:38 (2013). https://doi.org/10.1145/2483669.2483682
92. Yao, X., Mokbel, M.F., Alarabi, L., Eldawy, A., Yang, J., Yun, W., Li, L., Ye, S., Zhu, D.: Spatial coding-based approach for partitioning big spatial data in Hadoop. Computers & Geosciences **106**, 60–67 (2017). https://doi.org/10.1016/j.cageo.2017.05.014
93. Yoo, J.S., Shekhar, S.: A Joinless Approach for Mining Spatial Colocation Patterns. IEEE Transactions on Knowledge and Data Engineering **18**(10), 1323–1337 (2006). https://doi.org/10.1109/TKDE.2006.150
94. Yu, J., Wu, J., Sarwat, M.: GeoSpark: A Cluster Computing Framework for Processing Large-scale Spatial Data. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL '15, pp. 70:1–70:4. ACM, New York, NY, USA (2015). https://doi.org/10.1145/2820783.2820860
95. Zheng, Y.: Trajectory Data Mining: An Overview. ACM Trans. Intell. Syst. Technol. **6**(3), 29:1–29:41 (2015). https://doi.org/10.1145/2743025
96. Zhou, X., Shekhar, S., Ali, R.Y.: Spatiotemporal change footprint pattern discovery: an inter-disciplinary survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **4**(1), 1–23 (2014). https://doi.org/10.1002/widm.1113
97. Zhou, X., Shekhar, S., Mohan, P., Liess, S., Snyder, P.K.: Discovering interesting sub-paths in spatiotemporal datasets: A summary of results. In: Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems, pp. 44–53. ACM (2011)

Multimedia Data Learning

Zhongfei (Mark) Zhang and Ruofei (Bruce) Zhang

1 Introduction

Multimedia data learning (a.k.a. multimedia data mining), as the name suggests, presumably is a combination of the two existing areas: *multimedia* and *data mining*. However, multimedia data mining is *not* a research area that just simply combines the research of multimedia and data mining together. Instead, the multimedia data mining research focuses on the theme of merging multimedia and data mining research together to exploit the synergy between the two areas to promote the understanding and to advance the development of the knowledge discovery in multimedia data. Consequently, multimedia data mining exhibits itself as a unique and distinct research area that synergistically relies on the state-of-the-art research in multimedia and data mining as well as beyond (specifically and noticeably including machine learning and artificial intelligence in general) but at the same time fundamentally differs from either multimedia or data mining or a simple combination of the two areas.

Historically, multimedia and data mining are two very interdisciplinary and multidisciplinary areas. Both areas began in early 1990s with only a relatively short history. Therefore, both areas are relatively young areas (in comparison, for example, with many well established areas in computer science such as operating systems, programming languages, and artificial intelligence). On the other hand,

Z. (Mark) Zhang (✉)
Binghamton University, State University of New York, Binghamton, NY, USA
Microsoft AI & Research, Sunnyvale, CA, USA
e-mail: zzhang@binghamton.edu

R. (Bruce) Zhang
Google, Mountain View, CA, USA
e-mail: brzhang@google.com

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_19

with substantial application demands, both areas have undergone independently and simultaneously rapid developments in recent years.

Multimedia is a very diverse, interdisciplinary, and multidisciplinary research area.[1] The word *multimedia* refers to a combination of multiple media types together. Due to the advanced development of the computer and digital technologies in early 1990s, multimedia began to emerge as a research area [33, 78]. As a research area, multimedia refers to the study and development of an effective and efficient multimedia system targeting a specific application. In this regard, the research in multimedia covers a very wide spectrum of topics, ranging from multimedia indexing and retrieval, multimedia databases, multimedia networks, multimedia presentation, multimedia quality of services, multimedia usage and user study, to multimedia standards, just to name a few.

While the area of multimedia is so diverse with many different topics, only part of them is related to multimedia data mining, such as cross-media analysis, multimedia indexing and retrieval, multimedia databases, and multimedia presentation [29, 46, 79, 93, 94, 56]. Today, it is well-known that multimedia data are ubiquitous and are often required, if not necessarily essential, in many applications. This phenomenon has made multimedia repositories widespread and extremely large. There are tools for managing and searching within these collections, but the need for tools to extract hidden, unknown, potentially useful knowledge buried within multimedia collections is becoming pressing and central for many decision-making applications. For example, it is highly desirable for developing the tools needed today for discovering relationships between objects or segments within images, classifying images based on their content, extracting patterns in sound, categorizing speech and music, recognizing and tracking objects in video streams, and discovering and tracking events in social media.

At the same time, researchers in multimedia information systems, in the search for techniques for improving the indexing and retrieval of multimedia information, are looking for new methods for discovering effective indexing information. A variety of techniques, from machine learning, statistics, database, knowledge acquisition, data visualization, image analysis, high performance computing, and knowledge-based systems, have been used mainly as research handcraft activities. The development of multimedia databases and their query interfaces recalls again the idea of incorporating multimedia data mining methods for effective dynamic indexing.

On the other hand, data mining is also a very diverse, interdisciplinary, and multidisciplinary research area. The terminology *data mining* refers to knowledge discovery from data, presumably and, in particular, today massive, diverse, and noisy data collections. Originally, this area began with knowledge discovery in

[1] Here we are only concerned with a research area; multimedia may also be referred to industries and even social or societal activities.

databases. However, data mining research today has been advanced far beyond the area of databases [28, 39]. This is due to the following two reasons. First, today's knowledge discovery research requires more than ever the advanced tools and theories beyond the traditional database area, noticeably including mathematics, statistics, machine learning, and pattern recognition. Second, with the fast explosion of the data in scale and variety, and also the typical presence of multimedia data almost everywhere, it is not enough for today's knowledge discovery research to just focus on the structured data in the traditional databases; instead, it is common to see that the traditional databases have evolved into data warehouses, and the traditional structured data have evolved into more non-structured data such as imagery data, time-series data, spatial data, video data, audio data, and more general multimedia data as well as relational data. Adding into this complexity is the fact that in many applications these non-structured data do not even exist in a more traditional "database" anymore; they are just simply a collection of the data, even though many times people still call them databases (e.g., image database, video database).

Examples are the data collected in fields such as art, design, hypermedia and digital media production, case-based reasoning, and computational modeling of creativity, including evolutionary computation, and medical multimedia data. These exotic fields use a variety of data sources and structures, interrelated by the nature of the phenomenon that these structures describe. As a result there is an increasing interest in new techniques and tools that can detect and discover patterns that lead to new knowledge in the problem domain where the data have been collected. There is also an increasing interest in the analysis of multimedia data generated by different distributed applications, such as collaborative virtual environments, virtual communities, and multi-agent systems. The data collected from such environments include a record of the actions in them, a variety of documents that are part of the business process, asynchronous threaded discussions, transcripts from synchronous communications, and other data records. These heterogeneous multimedia data records require sophisticated preprocessing, synchronization, and other transformation procedures before even moving to the analysis stage.

Consequently, with the independent and advanced developments of the two areas of multimedia and data mining, with today's explosion of the data scale and the existence of the pluralism of the data media types and varieties, it is natural to evolve into this new area called *multimedia data mining*. While it is presumably true that multimedia data mining is a combination of the research between multimedia and data mining, the research in multimedia data mining refers to the synergistic applications of knowledge discovery theories and techniques in a multimedia collection. As a result, "inherited" from its two parent areas of multimedia and data mining, multimedia data mining by nature is also an interdisciplinary and multidisciplinary area; in addition to the two parent areas, multimedia data mining also relies on the research from many other areas, noticeably including mathematics, statistics, machine learning, computer vision, and pattern recognition. Figure 1 illustrates the relationships among these interconnected areas.

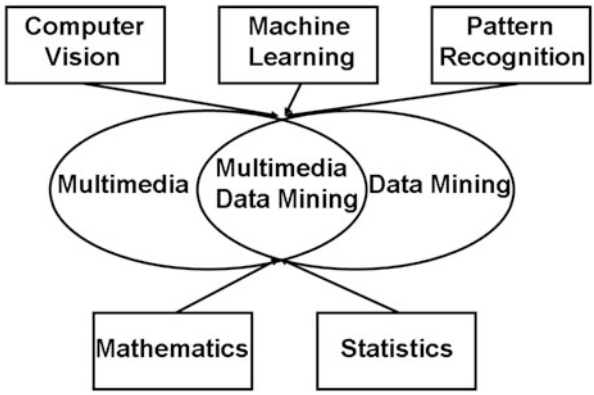


Fig. 1 Relationships among the interconnected areas to multimedia data mining. Redrawn from [94]

While we have clearly given the working definition of multimedia data mining as an emerging, active research area, due to historic reasons, it is helpful to clarify several misconceptions and to point out several pitfalls at the beginning.

- *Multimedia Indexing and Retrieval* vs. *Multimedia Data Mining*: It is well-known that in the classic data mining research, the pure text retrieval or the classic information retrieval is *not* considered as part of data mining, as there is no knowledge discovery involved. However, in multimedia data mining, when it comes to the scenarios of multimedia indexing and retrieval, this boundary becomes vague. The reason is that a typical multimedia indexing and/or retrieval system reported in the recent literature often contains a certain level of knowledge discovery such as feature selection, dimensionality reduction, concept discovery, as well as mapping discovery between different modalities (e.g., imagery annotation where a mapping from an image to textual words is discovered and word-to-image retrieval where a mapping from a textual word to images is discovered). In this case, multimedia information indexing and/or retrieval is considered as part of multimedia data mining. On the other hand, if a multimedia indexing or retrieval system uses a "pure" indexing system such as the text-based indexing technology employed in many commercial imagery/video/audio retrieval systems on the Web, this system is not considered as a multimedia data mining system as there is no knowledge discovery as stated above.
- *Database* vs. *Data Collection*: In a classic database system, there is always a database management system to "govern" all the data in the database. This is true for the classic, structured data in the traditional databases. However, when the data become non-structured data, in particular, multimedia data, often we do not have such a management system to "govern" all the data in the collection. Typically, we simply just have a whole collection of multimedia data, and we expect to develop an indexing/retrieval system or other data mining system on top of this data collection. For historic reasons, in many literature references,

people still use the terminology of "database" to refer to such a multimedia data collection, even though this is different from the traditional, structured database in concept.

- *Multimedia Data* vs. *Single Modality Data*: Although "multimedia" refers to the multiple modalities and/or multiple media types of data, conventionally in the area of multimedia, multimedia indexing and retrieval also includes the indexing and retrieval of a single, non-textual modality of data, such as image indexing and retrieval, video indexing and retrieval, and audio indexing and retrieval. Consequently, in multimedia data mining, we follow this convention to include the study of any knowledge discovery dedicated to any single modality of data as part of the multimedia data mining research. Therefore, studies in image data mining, video data mining, and audio data mining as well as text data mining alone are considered as part of the multimedia data mining area.

Multimedia data mining, although still in its early booming stage as an area that is expected to have further development, has already found enormous application potential in a wide spectrum covering almost all the sectors of society, ranging from people's daily life to economic development to government services. This is due to the fact that in today's society almost all the real-world applications often have data with multiple modalities, from multiple sources, and in multiple formats. For example, in homeland security applications, we may need to mine data from an air traveler's credit history, traveling patterns, photo pictures, and video data from surveillance cameras in the airport. In the manufacturing domains, business processes can be improved if, for example, part drawings, part descriptions, and part flow can be mined in an integrated way instead of separately. In medicine, a disease might be predicted more accurately if the MRI (magnetic resonance imaging) imagery is mined together with other information about the patient's condition. Similarly, in bioinformatics, data are available in multiple formats.

The rest of the chapter is organized as follows. In the next section, we give the architecture for a typical multimedia data mining system or methodology in the literature. We then describe in detail the key components of such a system. Finally, we conclude this chapter with a note for further readings.

2 A Typical Architecture of a Multimedia Data Mining System

A typical multimedia data mining system, or framework, or method always consists of the following three key components. Given the raw multimedia data, the very first component for mining the multimedia data is to convert a specific raw data collection into a representation in an abstract space which is called the *feature space*. This process is called *feature extraction* or *feature learning*. Consequently,

we need a feature representation method to convert the raw multimedia data to the features in the feature space, before any mining activities are able to be conducted. This component is extremely important as the success of a multimedia data mining system to a large degree depends upon how good the feature representation method is. The typical feature representation methods or techniques are taken from the classic machine learning research, computer vision research, pattern recognition research, as well as multimedia information indexing and retrieval research in multimedia area.

Since knowledge discovery is an intelligent activity, like other types of intelligent activities, multimedia data mining requires the support of a certain level of knowledge. Therefore, the second key component is the knowledge representation, i.e., how to effectively represent the required knowledge to support the expected knowledge discovery activities in a multimedia data mining system. The typical knowledge representation methods used in the multimedia data mining literature are directly taken from the general knowledge representation research in artificial intelligence area with the possible special consideration in the multimedia data mining problems.

Finally, we come to the last key component—the actual mining or learning theory and/or technique to be used for the knowledge discovery in a multimedia data mining system. In the current literature of multimedia data mining, there are mainly two paradigms of the learning or mining theories/techniques that can be used separately or jointly in a specific multimedia data mining application. They are *statistical learning theories* and *soft computing theories*, respectively. The former is based on the recent literature on machine learning and, in particular, statistical machine learning, whereas the latter is based on the recent literature on soft computing and, in particular, deep learning. This component typically is the core of the multimedia data mining system.

In addition to the three key components, in many multimedia data mining systems, there are user interfaces to facilitate the communications between the users and the mining systems. Like the general data mining systems, for a typical multimedia data mining system, the quality of the final mining results can only be judged by the users. Hence, it is necessary in many cases to have a user interface to allow the communications between the users and the mining systems and the evaluations of the final mining quality; if the quality is not acceptable, the users may need to use the interface to tune different parameter values of a specific component used in the system, or even to change different instantiations of the component, in order to achieve better mining results, which may go into an iterative process until the users are happy with the mining results.

Figure 2 illustrates this typical architecture of a multimedia data mining system. We elaborate each of the three key components in the next three sections.

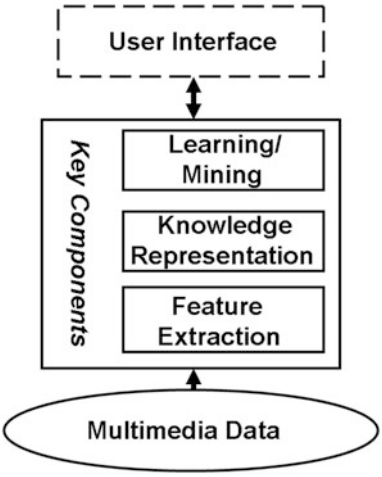


Fig. 2 The typical architecture of a multimedia data mining system. Redrawn from [94]

3 Feature Extraction

In general, features are the abstraction of the data in a specific modality defined in measurable quantities in a specific Euclidean space [32]. The Euclidean space is thus called *feature space*. Features, also called *attributes*, are an abstract description of the original multimedia data in the feature space. Since typically there are more than one feature used to describe the data, these multiple features form a *feature vector* in the feature space. The process of identifying the feature vector from the original multimedia data is called *feature extraction*. Depending upon different features defined in a multimedia system, different feature extraction methods are used to obtain these features.

Typically, features are defined with respect to a specific modality of the multimedia data. Consequently, given multiple modalities of multimedia data, we may use a feature vector to describe the data in each modality. As a result, we may use a combined feature vector for all the different modalities of the data (e.g., a concatenation of all the feature vectors for different modalities) if the mining is to be performed in the whole data collection aggregatively, or we may leave the individual feature vectors for the individual modalities of the data if the mining is to be performed for different modalities of the data separately.

In the classic literature, feature extraction refers to applying one of the existing feature extraction methods to the raw multimedia data to obtain the resulting features of the raw data, where the existing feature extraction methods are all well-defined in advance through manual feature engineering. Recently, methods are developed to obtain the features *automatically*, as opposed to through a manual

feature engineering, which is called *feature learning* or *representation learning* [9]. Below we first review the classic feature extraction methods. Then we give a brief review on feature learning.

We note that there are several commonly encountered media types or modalities in multimedia data mining. They can be represented in terms of the dimensions of the space the data are in. Specifically, we list those commonly encountered media types or modalities as follows.

- *0-dimensional data*: This type of the data is the regular, alphanumeric data. A typical example is the text data.
- *1-dimensional data*: This type of the data has one dimension of a space imposed into them. A typical example is the audio data.
- *2-dimensional data*: This type of the data has two dimensions of a space imposed into them. Imagery data and graphics data are the two common examples.
- *3-dimensional data*: This type of the data has three dimensions of a space imposed into them. Video data and animation data are the two common examples.

Correspondingly, there are several media-specific, classic feature extraction methods that we briefly introduce below.

- *TF-IDF*: The TF-IDF measure is specifically defined as a feature for text data. Given a text collection of N documents and a total M word vocabulary, the standard text processing model is based on the *bag-of-words* (BOW) assumption, which says that for all the documents, we do not consider any linguistic or spatial relationships among the words in a document; instead, we consider each document just as a collection of isolated words, resulting in a bag-of-words representation. Given this assumption, we represent the collection as an $N \times M$ matrix which is called the *Term Frequency Matrix*, where each entry $TF(i, j)$ is the occurrence frequency of the word j occurring in the document i. Therefore, the total term frequency for the word j is

$$TF(j) = \sum_{i=1}^{N} TF(i, j) \tag{1}$$

  In order to penalize those words that appear too frequently, which does not help in indexing the documents, an *inverse document frequency* (IDF) is defined as

$$IDF(j) = \log \frac{N}{DF(j)} \tag{2}$$

  where $DF(j)$ means the number of the documents in which the word j appears and is called the *document frequency* for the word j. Finally, TF-IDF for a word j is defined as

$$\text{TF-IDF}(j) = TF(j) \times IDF(j) \tag{3}$$

The details of the TF-IDF feature may be found in [70].

- *Cepstrum*: Cepstrum features are often used for one-dimensional media type data such as audio data. Given such data represented as a one-dimensional signal, *cepstrum* is defined as the Fourier transform of the signal's decibel spectrum. Since the decibel spectrum of a signal is obtained by taking the logarithm of the Fourier transform of the original signal, cepstrum is sometimes in the literature also called the *spectrum of a spectrum*. The technical details of the cepstral features may be found in [18].
- *Fundamental Frequency*: This refers to the lowest frequency in a series of harmonics a typical audio sound has. If we represent the audio sound in terms of a series of sinusoidal functions, the fundamental frequency refers to the frequency that the sinusoidal function with the lowest frequency in the spectrum has. Fundamental frequency is often used as a feature for audio data mining.
- *Audio Sound Attributes*: Typical audio sound attributes include pitch, loudness, and timbre. *Pitch* refers to the sensation of the "altitude" or the "height," often related to the frequency of the sounds, and, in particular, related to the fundamental frequency of the sounds. *Loudness* refers to the sensation of the "strength" or the "intensity" of the sound tone, often related to the sound energy intensity (i.e., the energy flow or the oscillation amplitude of the sound wave reaching the human ear). *Timbre* refers to the sensation of the "quality" of the audio sounds, often related to the spectrum of the audio sounds. The details of these attributes may be found in [78]. These attributes are often used as part of the features for audio data mining.
- *Optical Flow*: Optical flows are the features often used for three-dimensional media type data such as video and animation. Optical flows are defined as the changes of an image's brightness of a specific location of an image over the time in the motion picture streams. A related but different concept is called *motion field*, which is defined as the motion of a physical object in a three-dimensional space measured at a specific point on the surface of this object mapped to a corresponding point in a two-dimensional image over the time. Motion field represented by 2D vectors is useful information in recovering the three-dimensional motion from an image sequence in computer vision research [47]. Since there is no direct way to measure the motion vectors in an image plane, often it is assumed that the motion vectors are the same as the optical flows and thus the optical flows are used as the motion vectors. However, conceptually they are different. For the details of the optical flows as well as their relationship to the motion vectors, see [44].

Essentially, in the classic feature extraction literature, there are three categories of features that are often used in the literature. They are *statistical features*, *geometric features*, and *meta features*. Except for some of the meta features, most of the feature representation methods are applied to a *unit* of multimedia data instead of to the whole multimedia data, or even to a part of a multimedia data unit. A unit

of multimedia data is typically defined with respect to a specific modality of the data. For example, for an audio stream, a unit is an audio frame; for an imagery collection, a unit is an image; for a video stream, a unit is a video frame. A part of a multimedia data unit is called an *object*. An object is obtained by a segmentation of the multimedia data unit. In this sense, the feature extraction is a mapping from a multimedia data unit or an object to a feature vector in a feature space. We say that a feature is *unique* if and only if different multimedia data units or different objects map to different values of the feature; in other words, the mapping is one-to-one. However, when this uniqueness definition of features is carried out to the object level instead of the multimedia data unit level, different objects are interpreted in terms of different *semantic objects* as opposed to different variations of the same object. For example, an apple and an orange are two different semantic objects, while different views of the same apple are different variations of the same object but not different semantic objects.

Below we review several well-known classic feature representation methods in each of the categories.

Statistical features focus on a statistical description of the original multimedia data in terms of a specific aspect such as the frequency counts for each of the values of a specific quantity of the data. Consequently, all the statistical features only give an aggregate, statistical description of the original data in an *aspect*, and therefore, it is in general not possible to expect to recover the original information from this aggregate, statistical description. In other words, statistical features are typically *not* unique; if we conceptualize obtaining the statistical features from the original data as a transformation, this transformation is, in general, lossy. Unlike geometric features, statistical features are typically applied to the whole multimedia data unit without segmentation of the unit into identified parts (such as an object) instead of to the parts. Due to this reason, in general all the variation-invariant properties (e.g., translation-invariant, rotation-invariant, scale-invariant, or the more general affine-invariant) for any segmented part of a multimedia data unit do not hold true for statistical features.

Well-known classic statistical features include histograms [27], transformation coefficients (e.g., Wavelet features [37]), coherent vectors [63], and correlograms [45].

Unlike statistical features, geometric features are typically applied to segmented or identified objects within a multimedia data unit of a specific modality of the data. Consequently, a segmentation method must be first applied to a multimedia data unit to obtain such objects. Once such objects are obtained, geometric features are used to describe these objects. Due to this purpose, many geometric features are variation-invariant, offering the capability to preserve the same description while the objects are subject to different variations such as rotation changes, translation changes, and scale changes from one unit to another.

Depending upon how "completely" a specific geometric feature method is capable of describing an object in a multimedia data unit, some of the geometric features are unique, while others are not. Well-known geometric features include

moments [27], inertia coefficients [34], Fourier descriptors [27] (including Fourier Area Descriptors [92]), and SIFT (Scale-Invariant Feature Transform) [57].

Finally, meta features include the typical meta data to describe a multimedia data unit such as the scale of the unit, the number of objects in the unit, and the value range of the points in the unit.

Recently, representation learning or feature learning [9] has become one of the hot research topics in machine learning, where the focus is on developing automatic methods to extract features from the raw data. In comparison with the classic feature extraction methods based on manual feature engineering, there are at least three advantages for feature learning. First, with feature learning we can completely get rid of the hassle of manual feature engineering that is typically tedious, ad hoc and would require substantial insight and/or a priori experience to either define a new feature extraction method or to select a specific one from the wide pool of the existing methods for a given problem; even if we have a feature extraction method ready for a given problem, we are still uncertain whether this method would lead to an optimal solution from feature extraction perspective, and thus would still need to try different existing ones in order to achieve a better performance. Second, feature learning is data-adaptive for the raw data in a specific problem, that is not available and also not possible for manual feature engineering in the classic feature extraction. Third, feature learning allows hierarchical feature extractions, which is consistent to a certain degree with human cognition and has been proven to be powerful and effective in delivering solutions to many real-world problems. Bengio et al. [9] give an extensive review on the recent representation learning literature. Specifically, Langkvist et al. [52] provide an extensive review on feature learning for time-series data and Li et al. [56] give an extensive review on feature learning for multi-view data. Below we briefly review several popular methods for feature learning.

Autoencoder An autoencoder is a multi-layer artificial neural network with an odd total number of layers that is symmetric with respect to the middle layer in terms of the network topology between the two sides of this middle layer such that from the input layer the number of nodes at each layer decreases towards the middle layer and increases from the middle layer in order to satisfy the constraint that the output reconstructs the input. Figure 3 shows a shallow autoencoder and usually the number

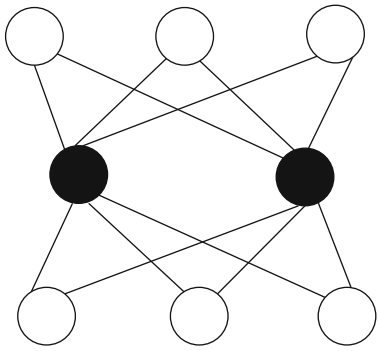


Fig. 3 A shallow autoencoder where there is only one hidden layer; a deeper autoencoder can be developed with more hidden layers to keep the symmetric structure with respect to the middle layer; here solid circles denote hidden nodes and open circles denote observable nodes such as input and output nodes

of hidden layers can be more than one with a deep network. Mathematically, given an input vector $\boldsymbol{x}$ and the output $\boldsymbol{y}$, an autoencoder $A(\boldsymbol{x}, \boldsymbol{\theta})$) must satisfy

$$\boldsymbol{y} = A(\boldsymbol{x}, \boldsymbol{\theta}) \quad s.t. \quad \boldsymbol{y} \approx \boldsymbol{x} \tag{4}$$

Hinton and Salakhutdinov [42] reported that with such an autoencoder we can achieve the purpose of dimensionality reduction for any input data $\boldsymbol{x}$ such that the output of the middle layer becomes the extracted feature vector for the input data $\boldsymbol{x}$. Further, they have shown that the obtained reduced dimensional features are much better than what is obtained by principal component analysis [73]. Consequently, autoencoders can be used to automatically extract features from multimedia data and further with a deep autoencoder a hierarchical feature representation can be obtained by taking the outputs from different hidden layers of this deep autoencoder.

Convolutional Neural Networks (CNNs) Convolutional neural networks (CNNs) are artificial feedforward neural networks with the convolution operator [67] as the fundamental receptive field operator in the networks. Thus, a CNN uses convolutional layers with the convolution operators to extract features of the input data. Further, we can generate a hierarchy of features of the input data by applying multiple convolutional layers in the network, resulting in a deep network. Since convolution operators expand the dimensions of the original data (e.g., an image of $p \times p$ dimensions becomes an image of $(p + 1) \times (p + 1)$ dimensions after applying one layer of convolution with a convolution kernel of 3 by 3), to prevent the data dimensions from keeping expanding, typically we apply a pooling layer after each convolutional layer. A pooling operator is a subsampling operator (e.g., for every 2 by 2 window of the data after applying a convolutional layer we take only one sample to replace the whole 2 by 2 window), and thus in the early literature it was also called subsampling (and correspondingly the layer was called subsampling layer). The reason for subsampling or pooling is that the specific location information after convolution does not need to be that "accurate" and also that we expect to keep in the convolution "invariant" at a higher, coarser level. There are different principles for selecting a specific data sample within a given subsampling window, such as taking the maximum (called max pooling), taking the average (called average pooling), and taking a data sample randomly (called stochastic pooling).

CNN is part of the foundation of today's deep learning architectures as it is extensively used as one of the key building blocks of the deep neural networks now [74, 80, 41]. The idea of CNN is not new, as it was studied and used for object recognition in the early literature of machine learning [53, 54, 11, 55]. However, the first deep neural network using CNN for large-scale object recognition is the well-known AlexNet [50] that consists of hierarchical five convolution-pooling layer pairs followed by two fully connected layers with 12 layers and 60 million parameters and 500,000 neurons in total to win the 1000-class ImageNet competition in 2012. After that, a number of even larger and deeper network architectures are developed in the literature based on CNN (such as VGG [74],

GoogleNet [80], and ResNet [41]) to continue pushing the horizon to advance the performances of the deep learning networks.

On the other hand, studies have been carried out in the literature on improving the performance and complexity of the existing CNN. For example, it was observed that there are repetitions of the filters subject to translations and rotations generated by a CNN layer (and, in particular, by the lower CNN layers in a deep network). These repetitions not only attribute to a substantial amount of unnecessary computational complexity but also hurt the overall performance. Consequently, more effective and efficient convolutional structures are developed in the literature to aim to remove these repetitions such as the Doubly Convolutional Neural Networks (DCNNs) [90] that can be used as an independent convolutional layer to replace any existing convolutional layer in the current deep networks.

Embedding While originally as a mathematical terminology, in machine learning and data mining communities, embedding refers to a vectorized representation of typically non-structured data such that the dimensionality of the resulting vectors is much lower than that of the data in the original space or in the space of a straightforward representation of the data. For example, an image embedding method maps an image of $m \times n$ into a feature vector in a p dimensional space with $p \ll m \times n$; a graph embedding method maps a graph $G(V, E)$ with $\mid V \mid = n, \mid E \mid = m$ into a p dimensional vector space such that $p \ll m \times n$; a text embedding method maps a word, or a phrase, or a sentence, or even a document into a vector in a space with a dimensionality much less than the vocabulary size.

In the context of multimedia data, for all the non-textual data, all the feature extraction or feature learning methods we have discussed above are actually the embedding methods, as they all map the given multimedia data into feature vectors represented in a feature space. Below we discuss text embedding.

As discussed above, in the classic text processing literature, with the well-known vector model using Tf-IDF features [70], since this is a BOW approach, each sentence or each document is represented as a vector in a space with the dimensionality as the vocabulary size, which is typically large (e.g., English vocabulary size is in the order of million). Even if we apply the standard preprocessing techniques to the original text data [70] such as stop word, word stemming, and entity identification to remove those non-discriminative words (i.e., the stop words) or to combine different words together as the same word (i.e., those words sharing the same stem or the same entity), the resulting vocabulary size is still very large.

In the classic natural language processing (NLP) literature, N-gram [36] is a well-known modeling technique to estimate the probability of a sentence by multiplying the conditional probabilities of each word in the sentence given the previous N-1 words of that word. Along this line of thought, recently the skip-gram model [60] is proposed to predict the occurrence probability of any word in a sentence given a specific word in this sentence, regardless of whether the word to be predicted occurs before or after the given word in the sentence. Mathematically, each word in a vocabulary is represented in the original vector space model (e.g., in BOW approach each word can be represented as a one-hot vector where only the entry of

this word in the vocabulary is 1 and all the other entries in the vector are 0). Assume that the whole vocabulary size is K. We can then train a classifier through a neural network to predict a specific context word c in a sentence given a specific word w in this sentence. The network architecture can be an encoder-decoder pair somewhat similar to the autoencoder shown in Fig. 3 with the middle layer as the encoder output as an embedding vector for the given word w, but unlike an autoencoder the encoder-decoder pair does not need to be symmetric with respect to the middle layer. Thus, the simplest form can be a simple one hidden layer network with that hidden layer as the middle layer, and of course a deeper architecture can also do. Consequently, for any word w in the vocabulary, there is always a corresponding embedding vector $\boldsymbol{x}_w \in R^d$ where the embedding space dimensionality $d \ll K$. Note that given this similarity to an autoencoder, the output of this architecture also regresses to a K-dimensional vector predicting the probabilities of the words in the vocabulary to be the context in a sentence given w. Hence, let $E(.)$ and $D(.)$ be the encoder and decoder, respectively. Let $\boldsymbol{x}_w = E(w)$, i.e., the embedding of word w. For any word c, the probability that c appears as a context word in a sentence given w is

$$P(c \mid w) = \frac{e^{\boldsymbol{x}_w{}^T D(\boldsymbol{x}_c)}}{\sum_{k=1}^{K} e^{\boldsymbol{x}_w{}^T D(\boldsymbol{x}_k)}} \tag{5}$$

where $\boldsymbol{x}_k$ is the embedding representation for any word k in the vocabulary. Equation 5 represents a multinomial regression with softmax activation.

Skip-gram is one of the whole suite of techniques called Word2Vec that map individual words into their corresponding vectors represented in a much lower dimensional space than that in the original word vector model space. The other well-known Word2Vec embedding method is the GloVe method [65]. Note that the similar embedding ideas can be applied not only to words but also to phrases, sentences, or documents, or even characters (e.g., Zhang et al. [91] shows that character level CNN works better than the word level in text classification in their empirical studies).

4 Knowledge Representation

In order to effectively learn the multimedia data, it is important that not only an appropriate feature representation is used for the multimedia data but also that appropriate knowledge support is available in a multimedia data mining system to facilitate the learning and reasoning tasks. Like all other intelligent systems, a typical multimedia data mining system is often equipped with a knowledge support component to "guide" the mining tasks. Typical types of knowledge in the knowledge support component include the domain knowledge, the common sense knowledge, as well as the meta knowledge. Consequently, how to appropriately and

effectively represent these types of knowledge in a multimedia data mining system has become a non-trivial research topic. On the other hand, like the general data mining activities, a typical multimedia data mining task is to automatically discover the knowledge in a specific context. Thus, there is also a knowledge representation problem after the knowledge is discovered in the mining activities.

Knowledge representation has been one of the active core areas in artificial intelligence research, and many specific representation methods are proposed in the literature [15]. In the rest of this section, we briefly review several well-known, classic knowledge representation methods with a conclusion for the recent developments. For a more detailed and specific review and description, refer to [94].

Logic Representation For human beings, a natural way to represent knowledge is through natural language. For example, in an imagery data mining system, if we intend to restrict the domain to the natural scenery, we may want to have the following specific piece of knowledge: *All the blue areas indicate either sky or water*. This piece of knowledge would help index images with either sky or water and is helpful to knowledge discovery such as an answer to the question *What is considered the typical scene in the images of the database with a blue sky?*. Given the fact that NLP is still an open area, an effective way to make the natural language understandable to a computer system is to use logic representation. A commonly used logic is called *predicate logic*, and the most popular predicate logic used in the literature is called *propositional first order logic*, typically abbreviated as FOL.

In FOL, all the variables are *set variables* in the sense that they can take any one of the values of the set defined for this variable. For example, a variable x may be defined as a real set [0, 1], which means that x may be any value in the domain of [0, 1]; or the variable x may be defined as a *color set* variable, in which we may explicitly define $x = \{\text{red}, \text{blue}, \text{white}\}$ and x may take any of the three possible colors. All the functions in FOL are called *predicates*. All the predicates in FOL are *Boolean predicates* in the sense that they can only return one of the two values: 0 for *false* and 1 for *true*. In addition, there are three operators defined for all the variables as well as the predicates. $\neg$ is a unitary operator applied to either a variable or a predicate resulting in a negation of the value of the operand. $\wedge$ is a binary operator applied to either variables or predicates for a multiplication between the values of the two operands. $\vee$ is another binary operator applied to either variables or predicates taking an addition between the two values of the two operands. Finally, there are two quantifiers defined for variables *only*, but not for predicates. They are the *universal quantifier* $\forall$ meaning *for all the values* of the variable to which this quantifier applies and the *existential quantifier* $\exists$ meaning that *there exists at least one value* of the variable to which this quantifier applies.

Semantic Networks Semantic networks (also called semantic graphs or knowledge graphs in the literature) are one very powerful knowledge representation tool used in today's artificial intelligence research and applications [81, 69, 16]. They are proposed in the literature to use graphs to represent the concepts and their relationships, in which the nodes in the graphs are the concepts and the edges in the graphs are the relationships. Historically, they were used to represent the English

words as well as their relationships in natural language understanding research in artificial intelligence [69]. Semantic networks represent loose associations between concepts. However, when they are used with added logical descriptions, semantic networks may have even more expressive power than FOL. Examples of these extensions of the semantic networks include the existential graphs [49] and the conceptual graphs [75].

Ontology Ontology is historically a subject in philosophy study, where the original focus is to study the concepts that have a direct relationship to being such as becoming, existence, and reality. In multimedia data mining research as well as the related areas including artificial intelligence and machine learning, ontology refers to a representation of the organization or categorization of the entities or objects or events of a domain or the world as well as their relationships among these entities/objects/events. Consequently, ontology is a special but popular case of semantic graphs.

In the classic research of knowledge representation and artificial intelligence, a specific ontology system is typically developed manually, which is typically small in scale and rather limited to a specific application problem. Today, taking the advantage of the Internet and big data, many large-scale ontology systems are developed by data-driven approaches. Examples of the well-known ontology systems include WordNet [4], ImageNet [5], Probase [1], Freebase [3], and Google Knowledge Graph [2].

Frames Frames are another knowledge representation method used in a multimedia learning system for describing a specific type of object or a specific abstract concept. This knowledge representation method was initially due to Minsky [61]. A frame may have a name as well as other attributes which have values; these attributes are also called *slots*. Concepts related to each other may be described for one frame as a slot value of another frame. For example, we may have the *house* frame to define and describe the related concepts, where the frame *house* has slots of *style*, *color*, *door*, etc., and some slots are described by next levels of the frame.

There is literature [40] arguing that frames have an equivalent expressive power in knowledge representation to that of logic. This is later demonstrated to be true to a certain extent [76].

Scripts Historically script is originally a classic psychological theory to study human behavior as typically human behavior can be modeled as a series of actions that may further be recursively modeled as another series of action. The modeling process can be described as a script of telling a story. In this classic theory, the fundamental unit of the behavior action is called a scene.

Schank and his colleagues applied this theory to develop this classic methodology for representing procedural or dynamic knowledge where a process or action or event is represented by a sequence of sub-processes or sub-actions recursively [71]. Consequently, scripts can be considered as a special case of frame representation where all the slots are defined in an order. An example of a script representation is the scenario of going to see a movie where the whole event is described by a series

of actions in order including purchasing tickets at the box office, going through admission, finding seats, sitting down in the seats, watching movies, and leaving movie theater.

Constraints A constraint is a condition or a set of conditions that constrains the description of an object or an event. In the classic artificial intelligence research, many problems are described through constraints, and therefore *constraint satisfaction* is considered as an effective approach to solving for those problems [69]. In multimedia data mining, typically there are three types of constraints: (1) *Attribute Constraints* define the characteristic description of a multimedia object or a multimedia data item or an event. (2) *Spatial Constraints* specify the spatial conditions that must hold true between multimedia data items or objects. (3) *Temporal Constraints* specify the temporal conditions that must hold true between multimedia data items or objects in an event.

At the methodological level, constraints may be easily represented in terms of FOL sentences. If we are restricted only to the constraints of unitary or binary variables, which is the most typical scenario in multimedia data mining, we may also use a *constraint graph* to represent a set of constraints where each node represents a variable and each edge represents a binary constraint between the two variables; the unitary constraints are represented as attributes for the related variable nodes in the graph.

Given a set of constraints, finding a feasible solution with the presence of such a set of constraints is called *constraint satisfaction* in artificial intelligence research. Solutions to constraint satisfaction are part of the artificial intelligence search methods. Typical methods for constraint satisfaction include *generate and test*, *backtracking*, *forward checking*, and *constraint propagation* [69]. Details of constraint satisfaction and the solutions may be found in [84].

Uncertainty Representation While all the knowledge representation methods introduced above are for "certain" knowledge, in many real-world multimedia data mining problems, there are many occasions in which there is uncertainty in the knowledge. Consequently, we need to study how to represent the uncertainty in the knowledge. In multimedia data mining, two commonly used approaches to representing uncertainties in knowledge are probability theory and fuzzy logic. Zhang and Zhang [94] gives detailed descriptions on these two approaches with examples.

Recent Developments As a key component in a typical multimedia data mining system as well as also in the general AI system, knowledge representation has been receiving extensive attention in the literature [77, 16] and a suite of practical knowledge base systems are developed [4, 5, 1, 3, 2]. For example, one of the important and also practical issue is with the extensive applications of deep learning techniques, which are typically end-to-end, how to incorporate a knowledge base into a deep learning system. Techniques such as memory networks [87, 51] are developed for addressing issues like this.

5 Learning Methodology

In the classic machine learning literature, from the beginning there have been always two schools existing: the symbolic computation school and the numerical computation school [62]. Historically, the symbolic computation school was considered as the dominant school in the classic literature of machine learning till after we had the Internet leading to the explosive generation of the data resulting in the Big Data era that was responsible for the rapid development of the statistical machine learning and, in particular, with the development of deep learning when the numerical computation school became dominant in the machine learning community. This is particularly true in the literature of multimedia data mining, partially due to the fact that almost all the multimedia sensors generate numerical data. Consequently, in the literature of multimedia learning methodology, almost all the methods are in the numerical computation school.

In the numerical computation school in the context of multimedia data learning, there are two families of the literature: Statistical Learning Based and Soft Computing Based.

5.1 Statistical Learning

There are many different statistical learning methods used to accommodate different multimedia data mining tasks. These methods not only require specific types of data structures but also imply certain types of algorithmic approaches. In the multimedia data mining context, the classification and regression tasks are especially pervasive, and the data-driven statistical machine learning theories and techniques are particularly important. Two major paradigms of statistical learning models that are extensively used in the recent multimedia data mining literature are the *generative learning models* and the *discriminative learning models*. Generative learning means that during the learning process a learner must actively construct or generate learning results through a learning hypothesis while discriminative learning means that the learning results are determined purely by the observations. Mathematically, in the context of supervised learning all the learning problems can be written down as learning a function $\boldsymbol{y} = f(\boldsymbol{x})$ with the observation $\boldsymbol{x}$ and the prediction or class label $\boldsymbol{y}$. In this context, discriminative learning is to learn this function through directly learning the posterior probability distribution $P(\boldsymbol{y} \mid \boldsymbol{x})$ given the observation $\boldsymbol{x}$ while generative learning is to learn this function through learning the joint probability distribution $P(\boldsymbol{x}, \boldsymbol{y})$.

Consequently, in the literature of statistical learning, examples of generative models include the Bayesian learning, ranging from the classic Naive Bayes Learning [94], to the Bayesian Belief Networks [19, 66, 22, 94], to the later developed graphical models including Latent Dirichlet Allocation (LDA) [12, 94], Probabilistic Latent Semantic Analysis (pLSA) [24, 43, 94], and Hierarchical

Dirichlet Process (HDP) [83, 94]. On the other hand, examples of discriminative models include the Support Vector Machines (SVMs) [20, 21, 14, 59, 94] and Support Vector Regression (SVR) [26, 94], as well as its development in the context of multimedia data learning on maximum margin learning with structured output space [23, 85, 82, 6, 38, 94], and the boosting theory for combining a series of weak classifiers into a stronger one [72, 30, 31, 94]. Considering the typical special application requirements in multimedia data mining where it is common that we encounter ambiguities and/or scarce training samples, two learning paradigms are also developed: multiple instance learning [25, 7, 58, 94] and semi-supervised learning [13, 95, 94] with their applications in multimedia data mining. The former addresses the training scenario when ambiguities are present, while the latter addresses the training scenario when there are only a few training samples available. Both scenarios are very common in multimedia data mining and, therefore, it is important to study these two learning paradigms in the context of multimedia data mining.

Zhang and Zhang [94] gives an extensive and detailed introduction to different methods of statistical learning as well as their relationships and applications to multimedia data mining.

5.2 Soft Computing Based Learning

In many multimedia data mining applications, it is often required to make a decision in an imprecise and uncertain environment. For example, in the application of mining an image collection with a query image of green trees, given an image in the data collection that is about a pond with a bank of earth and a few green bushes, is this image considered as a match to the query? Certainly this image is not a perfect match to the query; but, on the other hand, it is also not an absolute mismatch to the query. Problems like this example, as well as many others, have intrinsic imprecision and uncertainty that cannot be ignored in decision-making. Traditional learning systems fail to solve such problems, as they attempt to use *Hard Computing* techniques. In contrast, a *Soft Computing* methodology implies cooperative activities rather than autonomous ones, resulting in the different computing paradigms such as fuzzy logic [88, 94], neural networks including the recently developed deep learning theories [50, 80, 74, 41, 94], genetic algorithms [35, 94], rough theory [64], and evolutionary computation [8]. Consequently, soft computing opens up a different research direction for problem solving that is difficult to achieve using traditional hard computing approaches.

Historically, Zadeh was considered responsible for the initial development of the soft computing family after his seminal work leading to the development of fuzzy logic and fuzzy set [88]. Indeed, all the other paradigms of soft computing are developed after Zadeh's fuzzy logic and fuzzy set theory [89], except for the theories of artificial neural networks, which rooted back to the early work on perceptron in 1950s [68].

It is also noted that part of the literature considers probabilistic reasoning as part of soft computing (e.g., [86]). However, we strongly believe that there are fundamental differences between the theories of probabilistic reasoning as well as the general statistical learning and those of soft computing. First, the theoretic foundations between the two families are different. While probabilistic reasoning as well as the whole statistical learning is based on the complete and sound classic theory, it is well-known that the theories behind soft computing are still relatively incomplete with several foundational open issues identified in the literature (e.g., [48]). Second, the application philosophies are different. While probabilities are often used to represent uncertainties in the literature, the end result of a probabilistic reasoning or inference is often hard, as opposed to soft. Take the simplest example of naive Bayes classifier. While probabilities are used to represent different classes, which appears to be "soft," the end classification result is hard, i.e., each data sample is only given one class label. In comparison, soft computing methods may generate a result in a soft nature (e.g., a data sample may be classified into several classes simultaneously with a membership value associated with each class). Similarly, soft-SVM [17] appears to be "soft," but the end result is hard and thus is still considered in the family of statistical learning.

Nevertheless, the boundary between the two families is not always clear. For example, graphical models are traditionally considered as part of statistical learning family. But with the recent development in deep learning, deep graphical models are developed in the literature [10] that bridge the two families well.

6 Conclusion and Further Readings

In this chapter we have given a high-level overview on multimedia data mining. Clearly, as a multidisciplinary and interdisciplinary area, it is not possible at all to be exhaustive. Consequently, the literature from all the related areas is considered relevant. For example, in machine learning area, the premier conferences include International Conference on Machine Learning (ICML), International Conference on Advances of Neural Information Processing Systems (NeurIPS), and International Conference on Learning Representation (ICLR); in multimedia area, the premier conferences include ACM Multimedia (ACM MM), IEEE International Conference on Multimedia and Expo (IEEE ICME), and ACM International Conference on Multimedia Information Retrieval (ACM ICMR); in computer vision area, the premier conferences include IEEE International Conference on Computer Vision (IEEE ICCV), IEEE International Conference on Computer Vision and Pattern Recognition (IEEE CVPR), and European Conference on Computer Vision (ECCV); in NLP, the premier conferences include the annual conference of the Association for Computational Linguistics (ACL) and the annual conference on Empirical Methods in Natural Language Processing (EMNLP); in AI area, the premier conferences include AAAI annual conference on Artificial Intelligence (AAAI) and International Joint Conference on Artificial Intelligence (IJCAI); and

in data mining area, the premier conferences include ACM International Conference on Knowledge Discovery and Data Mining (ACM KDD), IEEE International Conference on Data Mining (IEEE ICDM), and SIAM International Conference on Data Mining (SDM). Similarly, the premier journals of these areas are also considered as important sources of the related literature. In addition, a very good and important source of the literature is the arXiv collection.

References

1. http://haixun.olidu.com/probase/index.htm.
2. https://developers.google.com/knowledge-graph/.
3. https://en.wikipedia.org/wiki/freebase.
4. http://wordnet.princeton.edu/.
5. http://www.image-net.org/.
6. Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *Proc. ICML*, Washington DC, August 2003.
7. P. Auer. On learning from multi-instance examples: empirical evaluation of a theoretical approach. In *Proc. ICML*, 1997.
8. T. Back, D.B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. 1997.
9. Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *arXiv:1206.5538v3*, April 2014.
10. Y. Bengio, I.J. Goodfellow, and A. Courville. *Deep Learning*. MIT Press, 2017.
11. Y. Bengio, Y. LeCun, and D. Henderson. Globally trained handwritten word recognizer using spatial representation, space displacement neural networks, and hidden Markov models. In *Proceedings of Advances in Neural Information Processing Systems*, 1994.
12. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, (3):993–1022, 2003.
13. A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. Workshop on Computational Learning Theory*. Morgan Kaufman Publishers, 1998.
14. B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *the 5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992.
15. R. Brachman and H. Levesque. *Readings in Knowledge Representation*. Morgan Kaufman, 1985.
16. S. Cebiric, F. Goasdoue, H. Kondylakis, D. Kotzinos, I. Manolescu, and G. Troullinou. Summarizing semantic graphs: A survey. *The VLDB Journal*, 28(3):295–327, 2019.
17. D.-R. Chen, Q. Wu, Y. Wing, and D.-X. Zhou. Support vector machine soft margin classifiers: Error analysis. *Journal of Machine Learning Research*, 5:1143–1175, 2004.
18. D.G. Childers, D.P. Skinner, and R.C. Kemerait. The cepstrum: A guide to processing. *Proceedings of the IEEE*, 65(10):1428–1443, 1977.
19. G. Cooper. Computational complexity of probabilistic inference using Bayesian belief networks (research note). *Artificial Intelligence*, (42):393–405, 1990.
20. C. Cortes and V. Vapnik. Support vector machine. *Machine Learning*, 20(3):273–297, 1995.
21. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
22. P. Dagum and M. Luby. Approximating probabilistic reasoning in Bayesian belief networks is np-hard. *Artificial Intelligence*, 60(1):141–153, 1993.
23. H. III Daume and D. Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proc. ICML*, Bonn, Germany, August 2005.
24. S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of American Sociation of Information Science*, 41:391–407, 1990.

25. T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31 ?71, 1997.
26. H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems 9, NIPS 1996*, pages 156–161, 1997.
27. R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification (2nd ed.)*. John Wiley and Sons, 2001.
28. C. Faloutsos. *Searching Multimedia Databases by Content*. Kluwer Academic Publishers, 1996.
29. C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, 1994.
30. Y. Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, 1990.
31. Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, (55), 1997.
32. K. Fukunaga. *Introduction to Statistical Pattern Recognition (Second Edition)*. Academic Press, 1990.
33. B. Furht, editor. *Multimedia Systems and Techniques*. Kluwer Academic Publishers, 1996.
34. A. Gersho. Asymptotically optimum block quantization. *IEEE Trans. on Information Theory*, 25(4):373–380, 1979.
35. D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Professional, 1989.
36. J. Goodman. A bit of progress in language modeling. *arXiv:cs/0108005v1*, 2001.
37. A. Grossmann and J. Morlet. Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM Journal on Mathematical Analysis*, 15(4), 1984.
38. Z. Guo, Z. Zhang, E.P. Xing, and C. Faloutsos. Enhanced max margin learning on multimodal data mining in a multimedia database. In *Proc. ACM International Conference on Knowledge Discovery and Data Mining*, 2007.
39. J. Han and M. Kamber. *Data Mining — Concepts and Techniques*. Morgan Kaufmann, 2 edition, 2006.
40. P. Hayes. The logic of frames, 1979.
41. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.
42. G.E. Hinton and R.R. Salakhutdinov.
43. T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177C196, 2001.
44. B.K.P. Horn. *Robot Vision*. MIT Press and McGraw-Hill, 1986.
45. Jing Huang and S. Ravi Kumar et al. Image indexing using color correlograms. In *IEEE Int'l Conf. Computer Vision and Pattern Recognition Proceedings*, Puerto Rico, 1997.
46. R. Jain. Infoscopes: Multimedia information systems. In B. Furht, editor, *Multimedia Systems and Techniques*. Kluwer Academic Publishers, 1996.
47. R. Jain, R. Kasturi, and B.G. Schunck. *Machine Vision*. MIT Press and McGraw-Hill, 1995.
48. K. Kawaguchi, L.P. Kaelbling, and Y. Bengio. Generalization in deep learning. *arXiv:1710.05468v5*, 2019.
49. K.L. Ketner and H. Putnam. *Reasoning and the Logic of Things*. Harvard University Press, 1992.
50. A. Krizhevsky, I. Sutskever, and G.E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2012.
51. A. Kumar, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. *arXiv:1506.07285v5*, 2016.

52. M. Langkvist, L. Karlsson, and A. Loutifi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.
53. Y. LeCun. Generalization and network design strategies. *Univ. Toronto Computer Science Department Technical Report*, CRG-TR-89-4, 1989.
54. Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a backpropagation neural network. In *Proceedings of Advances in Neural Information Processing Systems*, 1990.
55. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of IEEE*, 86(11), Pages=2278-2324, year=1998).
56. Y. Li, M. Yang, and Z. Zhang. A survey of multi-view representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 2018.
57. D. Lowe. Object recognition from local scale-invariant features. In *Proc. IEEE International Conference on Computer Vision*, September 1999.
58. O. Maron and T. Lozano-Perez. A framework for multiple instance learning. In *Proc. NIPS*, 1998.
59. E. Mayoraz and E. Alpaydin. Support vector machines for multi-class classification. In *IWANN (2)*, pages 833–842, 1999.
60. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representation of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems*, 2013.
61. M. Minsky. A framework for representing knowledge. In P.H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, 1975.
62. S. Papert. One ai or many? In S.R. Graubard, editor, *The Artificial Intelligence Debate: False Starts, Real Foundations*. MIT Press, 1988.
63. G. Pass and R. Zabih. Histogram refinement for content-based image retrieval. In *IEEE Workshop on Applications of Computer Vision*, Sarasota, FL, December 1996.
64. Z. Pawlak. Rough sets. *International Journal of Parallel Programming*, 11(5):341–356, 1982.
65. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.
66. M. Pradham and P. Dagum. Optimal Monte Carlo estimation of belief network inference. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 446–453, 1996.
67. W.K. Pratt. *Introduction to Digital Image Processing*. CRC Press, 2013.
68. F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.
69. S. Russell and P. Norvig. *Artificial intelligence: A modern approach*. Prentice Hall, Upper Saddle River, NJ, 1995.
70. G. Salton. Developments in automatic text retrieval. *Science*, 253:974–979, 1991.
71. R.C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1990.
72. R. Schapire. Strength of weak learnability. *Journal of Machine Learning*, 5:197–227, 1990.
73. J. Shlens. A tutorial on principal component analysis. *arXiv:1404.1100v1*, 2014.
74. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representation*, 2015.
75. J.F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.
76. J.F. Sowa. *Knowledge Representation – Logical, Philosophical, and Computational Foundations*. Thomson Learning Publishers, 2000.
77. J.F. Sowa. *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufmann, 2014.
78. R. Steinmetz and K. Nahrstedt. *Multimedia Fundamentals — Media Coding and Content Processing*. Prentice-Hall PTR, 2002.
79. V.S. Subrahmanian. *Principles of Multimedia Database Systems*. Morgan Kaufmann, 1998.

80. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2015.
81. S.L. Tanimoto. *Elements of Artificial Intelligence Using Common LISP*. Computer Science Press, 1990.
82. B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Neural Information Processing Systems Conference*, 2003.
83. Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. Hierarchical Dirichlet process. *Journal of the American Statistical Association*, 2006.
84. E.P.K. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.
85. I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. ICML*, Banff, Canada, 2004.
86. E. Turunen, K. Raivio, and T. Mantere. Soft computing methods. In S. Pohjolainen, editor, *Mathematical Modeling*. Springer, 2016.
87. J. Weston, C. Chopra, and A. Bordes. Memory networks. In *Proceedings of International Conference on Learning Representation*, 2015.
88. L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.
89. L.A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 37(3):77–84, 1994.
90. S. Zhai, Y. Cheng, W. Lu, and Z. Zhang. Doubly convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2016.
91. X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Proceedings of Advances on Neural Information Processing Systems*, 2015.
92. Z. Zhang, R. Jing, and W. Gu. A new Fourier descriptor based on areas (AFD) and its applications in object recognition. In *Proc. of IEEE International Conference on Systems, Man, and Cybernetics*. International Academic Publishers, 1988.
93. Z. Zhang, F. Masseglia, R. Jain, and A. Del Bimbo. Editorial: Introduction to the special issue on multimedia data mining, 2008.
94. Z. Zhang and R. Zhang. *Multimedia Data Mining — A Systematic Introduction to Concepts and Theory*. Taylor & Francis, 2008.
95. X. Zhu. Semi-supervised learning literature survey. *Technical Report*, 1530, 2005.

Web Mining

Petar Ristoski

1 Introduction

The World Wide Web (WWW) [1] emerged in the early 1990s, providing means to publish and access documents online. The WWW is based on the Hypertext Transfer Protocol (HTTP), used to transfer hypermedia documents between servers and clients, and the Hypertext Markup Language (HTML), which is the standard markup language for creating documents to be viewed in a Web browser. This allows users and organizations to easily and instantly publish information and documents on the Web. Since the beginning, a vast amount of information has been published on the Web, covering any imaginable topic and domain. Currently, the Web contains several billions of Web pages and several hundreds of billions of links between those pages, and it continuously expands. To be able to cope with such a large amount of data, especially being able to identify relevant information in it, *Web mining* approaches are being used. Web mining is the process of performing data mining on the Web, including Web documents, Web graph, and Web usage data. The main goal of Web mining is to identify and extract relevant information and knowledge hidden in Web data. Web mining approaches follow the standard knowledge discovery process, which consists of five steps: data collection, preprocessing, transformation, pattern discovery, and pattern analysis [2]. Web mining is a multidisciplinary research area involving approaches and techniques from many other research areas, e.g., databases, data mining, machine learning, information retrieval, information extraction, natural language processing, statistics, etc.

Web mining is commonly divided into three sub-areas:

P. Ristoski (✉)
IBM Research, San Jose, CA, USA
e-mail: petar.ristoski@ibm.com

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_20

1. Web Content Mining: The process of mining the content of the Web documents in order to identify useful information, using data mining and machine learning. This commonly includes mining unstructured or semi-structured text documents, images, audio and video files extracted from Web documents.
2. Web Structure Mining: The process of mining the graph structure of the Web to identify structural and connection information about the nodes in the graph. This is usually done using graph theory and graph mining techniques.
3. Web Usage Mining: The process of mining the Web server logs, in order to extract patterns and information about the user interactions with the Web servers of one or more Web sites.

There are several textbooks [3, 4, 5] that cover Web mining in details, as well as several surveys [6, 7, 8, 9]. For surveys on Web content mining, we refer to [10, 11, 12, 13], surveys for Web structure mining can be found in [14, 15, 16], and surveys for Web usage mining can be found in [17, 18, 19, 20, 21].

An additional sub-area of Web mining is Semantic Web mining, which is concerned with the application of Web mining approaches for the Semantic Web. The Semantic Web is an extension of the standard Web in which data is structured using well established formats and technologies, making it machine-readable and machine-understandable [22]. For surveys on Semantic Web mining, we refer to [23, 24, 25, 26, 27, 28, 29].

The remaining of the chapter is organized as follows. Section 2 gives an introduction to the graph structure and properties of the Web. Section 3 gives an overview of approaches used for text mining and information extraction on the Web. Section 4 surveys approaches for Web structure mining for information retrieval and social network analysis. Section 5 gives an overview of Web usage mining and query log mining approaches. Finally, Sect. 6 gives an introduction to Semantic Web and Semantic Web Mining.

2 The Graph Structure of the Web

The Web is a network, or a directed graph, where the documents are the nodes and the hyperlinks between the documents are the edges of the graph. The graph structure carries important information about the Web documents and the information contained in them, thus the Web structure becomes crucial for Web mining. As shown by Hall et al. [30], the Web and the underlying Web graph are evolving throughout time. The Web started as a collection of documents, also known as “the Web of documents,” when in the 2000s the users started to be more involved and chaining the Web to “the Web of people,” evolving to the present state called “the Web of data and social networks.”

Throughout the years, many studies have tried to analyze the structure and the properties of the continuously evolving Web graph. One of the biggest challenges for analyzing the whole Web graph and Web mining, in general, is downloading the

whole Web. To do so, there are special systems called *Web crawlers*, also known as *robots* or *spiders*, which are able to download Web pages in bulk. The architecture of a Web crawler is quite simple, i.e., given a set of seed *Uniform Resource Locators* (URLs), the crawler downloads and indexes the corresponding Web pages, then extracts the outgoing hyperlinks from those pages and iteratively downloads the new Web pages. The process continues until no new hyperlinks are discovered. While the algorithm looks trivial, there are many challenges to build an efficient crawler that can capture a significant portion of the Web and keep it up to date [31]. Many different approaches and strategies have been proposed for broad crawling of the Web [31, 32, 33], as well as focused crawls [34, 35], which focus on downloading Web pages that cover a certain topic.

Several broad crawls have been used to perform in-depth analysis of the whole Web graph. The first analysis on the whole Web was performed by Broder et al. [36]. They used two AltaVista crawls from 1999, each with more than 200 million URLs and 1.5 billion links. The main finding of this work was that the Web is structured as a giant bow tie, with a Strongly Connected Core component (SCC) of 56 million pages in the middle, and two side components with 44 million pages each. The side components are called IN and OUT, where IN consists of pages that can reach the SCC but cannot be reached from it; while OUT consists of pages that are accessible from the SCC but do not link back to it. Furthermore, an additional component called TENDRILS contains pages that cannot reach the SCC and cannot be reached from the SCC. The analysis on the whole graph showed that the diameter of the central core (SCC) is at least 28, and that the diameter of the graph as a whole is over 500. The bow tie structure describes a macroscopic structure of the Web. Donato et al. [37] study the inner part of the bow tie structure, revealing that each component actually has a significantly different structure and propose replacing the bow tie structure with a daisy structure, where the IN and OUT components are attached like petals to the core SCC. A similar study by Jonathan et al. [38] suggest that the structure of the Web is more like a teapot than a bow tie, indicating that the IN component is much bigger than the OUT component. However, later research show that most of these findings heavily depend on the crawl and the crawling process [39, 40, 41]. Given these new findings Meusel et al. [41, 42] and Lehmberg et al. [43] perform analysis on a 2012 crawl from the Common Crawl Foundation,[1] which contained 3.5 billion Web pages and 128 billion links. The analysis confirmed the existence of a giant connected core component; however, there are different proportions of nodes that can reach or that can be reached from the giant component, suggesting that the bow tie structure is dependent on the crawling process and is not a structural property of the Web. Furthermore, the analysis shows that the graph has a diameter of at least 5282. Such studies confirm that the Web continuously evolves, and it is not trivial to analyze the whole structure of the Web graph but remains crucial for successful Web mining.

[1] http://commoncrawl.org/.

3 Web Content Mining

Web content mining approaches extract or mine useful information and knowledge from Web documents. Although there are means to publish structured data (see Sect. 6), most of the data published on the Web remains in unstructured format, i.e., text. To be able to extract useful knowledge from the vast amount of text data published on the Web, text mining approaches are used [44, 45, 46]. This includes: information retrieval, document classification, information extraction, text generation, text summarization, opinion mining and sentiment analysis. Web content mining includes image, video, and multimedia processing as well; however, the main focus of Web content mining is text mining, therefore in this chapter we focus only on text mining.

Information Retrieval Information retrieval is the study concerned with representing, searching, and manipulating large collections of text [49]. As mentioned in the introduction, one of the biggest challenges for consuming the Web is identifying the small pieces of relevant information in the vast amount of data published on the Web. To do so, Web search engines are being used. Search engines allow the users to enter a set of keywords or a query, for which the engine will return a ranked list of Web pages relevant for the query. Some of the most popular search engines are Google, Bing, Yahoo Search, Yandex, and Baidu. Web search engines can be seen as traditional information retrieval system working on Web data [47, 48].

Search engines consist of 3 main components: crawl, index, and search. Crawling is the process of downloading Web documents in bulk, as explained in Sect. 2. Indexing is the process of efficiently and effectively storing the crawled Web pages in a structure that would allow efficient retrieval of the Web pages. Searching is the process of matching the user query with the indexed Web pages and retrieving the matched results ordered by relevance.

Once all the Web documents are collected, before the documents are passed to the indexer, the documents go through a document pre-processor. This usually includes: lexical analysis (including tokenization and token normalization), token weighing, stopword removal, stemming, phrase processing, and hyperlinks processing [50, 51]. After this step, each Web document is represented as a set of search terms (tokens or a set of tokens) with meta data for each token. There are several information retrieval models or document representation techniques used, starting from the more trivial Boolean and the vector space models [52], to more sophisticated probabilistic and language models [53, 54]. Web documents are indexed in an index structure, such as *inverted index* structure. For every search term the inverted index contains a list of Web documents that contain that search term. This allows the search engine to quickly identify all documents that contain a term from a user query. For a survey of more sophisticated indexing techniques we refer to the survey by Gani et al. [55]. During the indexing, usually terms are assigned relevance weight, which indicates the importance of the term in the current document. One of the most used term weighing techniques is *term frequency—inverse document frequency* (tf-idf).

Once the index is built, the search engine can process user queries. Usually the user queries consist of one or several search terms. When the user query is received, the search engine first pre-processes the query using the same pipeline used to pre-process the documents for indexing. Then, for each of the search terms the relevant documents with their weights are retrieved from the inverted index. Finally, for each retrieved document a rank score is calculated using a relevance ranking function. Relevance ranking functions usually consist of 2 parts: (i) content document relevance score based on the matched terms, which depends on the used retrieval model; (ii) document relevance score based on the graph structure (see Sect. 4). There are many content relevance ranking functions, e.g., cosine similarity, BF25 [56], relevance based on language models [53, 57], or using machine learning for information retrieval, called *learning to rank* [58]. Learning to rank is the task to automatically construct a ranking machine learning model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance [59]. Commercial search engines use combinations of many relevance functions in order to get the best results. In many cases the user query might be very narrow and the search will not return any relevant results. To alleviate this problem, a *query expansion* approaches are used. Query expansion approaches modify the input query by adding additional search terms with similar semantic meaning [60, 61, 62].

While standard Web search engines deal only with keyword-based queries, with the advances of machine learning and knowledge representation, many systems now offer natural language *question answering*. In such systems, the users can ask a question in a natural language form, which the system then processes and matches to the internal knowledge, and produces an answer in natural language format. Many Web sites adapt such systems to develop *chatbots* and *conversational assistants* to ease the interaction with the users [63, 64]. With the rapid advancements of deep machine learning and knowledge graphs (see Sect. 6), such systems are becoming more popular [65, 66].

Document Classification Document classification is the task of sorting documents into a given set of categories. Categorizing Web documents into a predefined taxonomy made searching and browsing the Web easier [67]. The Open Directory Project (DMOZ) was the first Web topic directory that organized Web pages in a hierarchical ontology. Such categorization was used by search engines to improve the search results. While today such topic directories do not play big role in the search engine ranking, Web document classification is crucial for organizing and maintaining content on the Web. There are many supervised machine learning approaches for document classification, e.g., Naive Bayes, Support Vector Machines, Logistic Regression, Decision Trees, etc. With the advance of word embeddings (e.g., word2vec [68], GloVe [69]) and deep learning approaches (e.g., CNN [70], RNN [71], LSTM [72], BERT [73]) document classification achieves even better results.

When the set of categories is not known upfront, unsupervised methods can be used, such as *clustering* [74] and *topic modeling* [75].

Information Extraction Information extraction approaches automatically extract structured information from unstructured or semi-structured text. Some of the typical tasks in web information retrieval include *named entity recognition* (e.g., identifying mentions of people, places, organization, products, etc.) and *relationship extraction* between such named entities [76]. Named entity recognition is one of the most important tasks of natural language processing, and therefore there is a plethora of work in the literature [77, 78, 79]. Approaches range from simple techniques using external vocabularies and knowledge bases [80], unsupervised methods [81, 82], sophisticated supervised solutions with advanced feature engineering based on Hidden Markov Models [83], Conditional Random Fields [84] and Support Vector Machines [85], and lately advanced deep neural networks [86, 87, 88, 89, 79].

Once the named entities are identified in text, usually the next task is to identify the relation between them. Popular systems for relation extraction range from early solutions based on SVMs and tree kernels [90, 91, 92, 93, 94] to most recent ones exploiting neural architectures [95, 96, 97]. Many approaches exploit large knowledge bases for distant supervision [98, 99, 100, 101, 102], as well as human-in-the-loop approaches [103]. Furthermore, several works have tackled the problem of open information extraction [104, 105, 106] where the focus is on general understanding of text rather than a focused extraction of named entities and specific relations. Furthermore, recent deep neural network approaches solve both tasks of named entity recognition and relation extraction as a single task [107].

In most Web sites, the Web pages are dynamically generated using similar templates, using data from the same database. *Wrapper induction* approaches use supervised machine learning approaches to learn patterns and rules for generating dynamic Web pages in order to automatically extract structured information from them [108, 109].

Besides mining unstructured text from the Web, a lot of work goes into mining semi-structured data, such as lists and tables. WebTables was the first project to extract content tables from the whole Web [110, 111]. The project shows how to extract useful information from Web tables, such as entities, attributes, and relations between entities, which can be used in different applications. Several approaches have been introduced for parsing and semantically annotating Web tables [112, 113, 114], which then can be used for improved search [115], query table extension [116, 117, 118] and knowledge base completion [119, 120]. One survey of recent approaches for Web table mining can be found in [121].

Text Generation and Summarization While text generation was introduced in the early 1990s [122], it became more popular in the recent years with the advance of neural networks [123]. Currently, *generative adversarial neural networks* with reinforcement learning [122, 123, 124, 125] and transformer-based neural networks [126] show state-of-the-art results. Such systems have been successfully used in many applications: generating product descriptions [127], generating product reviews [128, 129], weather forecast [130], and many others. Such systems change the way new content is being created and published to the Web.

On the other hand, *text summarization* approaches try to shorten existing text documents. Such approaches generate a concise summary of large texts, focusing on the pieces of text that provide the most useful information without losing the overall meaning [131]. While in the past most of the approaches were based on topic words, frequencies, and latent semantic analysis, in the recent years *seq2seq* neural networks [132] have become state-of-the-art solution [133]. Such systems allow users to faster identify and consume relevant information on the Web.

Opinion Mining and Sentiment Analysis Web users use the available content from social media platforms, forums, and blogs for their decision making, e.g., the decision to buy a product offered on an e-shop heavily depends on the product reviews written by other users. Thus, it is crucial to be able to automatically identify the opinion and sentiments in user generated text on the Web. Since 2000, sentiment analysis has grown to be one of the most important research areas in natural language processing. Thus, plethora of approaches for opinion mining and sentiment analysis have been proposed [134, 135, 136].

Similar to opinion mining and sentiment analysis approaches, social media mining approaches [137] identify patterns and trends from textual data published on social media, which later can be used in different applications. The most common use of social media mining is for advertising, i.e., identifying the user's interests based on the content they generate to identify the best advertisements for them. Besides advertisement, social media mining has been used for identifying incidents and crisis in real time [138, 139, 140, 141], sports analysis, political analysis, trends, and more [142]. An interesting problem in social media mining is identifying fake news [143], which with the advance of generative neural networks becomes a real problem.

4 Web Structure Mining

As described in Sect. 2, the Web is a directed graph, which structure intrinsically carries important information about Web pages and the type and the quality of data in them. Using graph theory and graph mining approaches such information can be extracted and used in different applications, such as information retrieval and different social network analysis on the whole Web graph or separate Web sites.

Information Retrieval Early Web information retrieval systems used only the content Web data to retrieve the most relevant Web pages for a given user search query, as shown in Sect. 3. However, with the fast growth of the Web, the size of returned search results for any search query became rather large. Analyzing a large number of search results is not convenient for the users and it is costly. To cope with the large amount of Web pages, Web information retrieval systems started using the Web graph structure to identify the *popularity* of the Web pages. Such a popularity score is then used in the relevance ranking function in information retrieval systems. The popularity score of a Web page is calculated directly from

the Web graph structure, i.e., by examining how many hyperlinks point to the Web page and the popularity of the Web pages pointing to it. One naive solution is to use the number of incoming links to the Web page, also known as in-degree of the Web page, as a popularity score. However, this score can easily be manipulated by setting hyperlinks to Web pages that are not relevant, known as *spamming*, which would significantly decrease the performance of the search engines. In 1998 and 1999, the two most important Web page ranking algorithms were introduced, *PageRank* [144, 145] and *HITS* [146]. Both of these algorithms originate from social network analysis [147, 148].

The HITS (hyperlink-induced topic search) algorithm [146], also known as *hubs and authorities*, was introduced by Kleinberg in 1999. For a given search query, the algorithm identifies two types of Web pages: (i) *authorities* are pages that contain relevant information for the query; (ii) *hubs* are pages that contain hyperlinks to good sources. The main idea of HITS is that there is a mutual reinforcement relationship between authorities and hubs, i.e., good hubs link to many good authorities, and a good authority is linked to by many good hubs. Given a search query, the algorithm first extends the set of search result pages by adding all the pages that point to any of the pages in the result list, or pages that are pointed to by any of the pages in the result list. In the next step, each page in the expanded list is assigned authority score and a hub score. The authority score of a Web page is calculated as the sum of the hub scores of all Web pages pointing to it. The hub score of a Web page is calculated as the sum of the authority scores of all Web pages that the current page is pointing to. The scores are computed iteratively, and after each iteration the values are normalized between 0 and 1. Kleinberg proved that the algorithm will always converge within couple of iteration, which has been shown many times in practical experiments. While the algorithm has shown great performance to identify relevant pages for a given query, the algorithm has two main drawbacks that make it unusable in modern search engines: (i) it is sensitive to spamming, i.e., the hub score can be manipulated by adding a lot of outgoing links to good authorities; (ii) the scores are query specific and must be calculated during the search, which significantly increases the search time.

On the other hand, PageRank [144, 145] is a static Web page ranking algorithm, i.e., the PageRank value for each Web page in the Web graph is calculated offline and it is not query dependent. The PageRank algorithm was introduced by Brian and Page in 1988 and it is used in the popular *Google Search*. The underlying assumption of PageRank is similar to the one of HITS, i.e., more important Web pages are more likely to be linked to buy more important Web pages. The PageRank of a Web page is calculated as the sum of the PageRank values of all the Web pages pointing to it, where the PageRank value of each Web page is divided by the total number of links going out of that page. Furthermore the algorithm introduces a damping factor, which simulates the probability of a random surfer continuing to click on links as they browse the Web. Then the PageRank value of a Web page A is calculated as:

$$PR(A) = \frac{1-d}{N} + d \sum_{(A,B)} \frac{PR(B)}{O_B} \quad (1)$$

where d is the damping factor, which is set between 0 and 1, N is the total number of pages in the Web graph, and O_B is the number of outgoing links for any Web page B that links to A. The PageRank value is calculated iteratively.

The main strengths of the PageRank algorithm are being query independent and unaffected by spam. Although Google Search currently uses many other signals and features for ranking Web pages, it is noteworthy that even after many years of existence, PageRank is still being used in Google Search. There are several extensions of the PageRank algorithm. Xing and Ghorbani proposed the weighted PageRank algorithm [149], which takes into account the importance of both the incoming and the outgoing links of the pages and distributes rank scores based on the popularity of the pages. Li and Liu introduce the TS-Rank algorithm [150] to address the issue of assigning low PageRank scores to new Web pages, which might contain high quality content.

Social Network Analysis Social network analysis is the study of social entities, such as people, organizations, groups, Web pages, or any knowledge entities, and the relationships between them [147, 148]. Such analysis can infer interesting properties, roles, and relationships between nodes or group of nodes in the network. For example, HITS and PageRank are being used for identifying the popularity of different Web pages in the whole Web graph. An important social network study is *community detection* in networks [151]. A community represents a group of entities that cover a same topic, share a same interest, or are involved in an event. Identifying such communities can give interesting insights on the structure of the network and the function of different communities, which could be used in various applications, e.g., topic-based content clustering, recommender systems, advertising, etc. Several approaches have been proposed for community identification on the whole Web [152, 153], to identify different clusters of Web pages. However, most of the recent approaches focus on community detection in social media platforms on the Web, which connect millions of users and groups [154, 155].

With the advance of deep learning and representation learning in the recent years, network representation learning approaches are introduced [156]. Network representation learning approaches learn latent, low-dimensional representation of network nodes, while preserving the network structure and node content and attributes. Such representation of nodes in a network can be used in many social network analysis on the Web, such as classification [157, 158], link prediction [159], clustering [160], recommender systems [161], visualization [162], and search [163]. Several recent surveys give an overview of such approaches and their applications on different networks on the Web [156, 164, 165].

5 Web Usage Mining

Web usage mining is the process of identifying and analyzing patterns in Web server logs, including clickthrough, clickstream, user transactions, and other data about user interactions with the Web server, which are then used to improve the performance of different services on the Web [17, 166]. The main data source for Web usage mining are server logs, which include web server logs and application server logs. From such data, information about each visit and interaction with the Web server is recorded. The first step of web usage mining is processing the raw log files and extract structured information [167], which typically includes: (i) user identification—identifying all actions that belong to the same user; (ii) Pageview identification—identifying which Web pages are being visited including the attributes of the Web pages; (iii) Sessionization—identifying a set of pages visited by the same user over one visit to a Web site; (iv) Episode identification—identifying a set of Web pages visited by the user that are semantically or functionally related. In the next step, these data are transformed into a data model, which can be used in different data mining algorithms to identify useful patterns from the log files. There are several common mining approaches used for improving the content, structure, and design of Web sites, including session and user analysis, which gives basic visit and popularity statistics of different Web pages within a Web site; classification and clustering of users to identify user communities (see Sect. 4); identifying associated Web pages, or products and services that are commonly bought together, using association rule mining. More advanced applications of Web usage mining involve recommender systems on the Web, and applications of query log mining.

Recommender Systems Recommender systems are machine learning models that predict a rating score or a binary preference a user would give to an item [168]. Recommender systems are the core component of every e-commerce Web site, used for recommending products and services to users based on their preferences, interests and previous interactions with the Web site. Since the performance of the recommender systems directly affects the success and the profit of e-shops, plethora of approaches have been proposed in the literature [169, 170]. There are three general types of recommender systems: (i) *Collaborative filtering* methods recommend items that are liked/bought by users with similar interests and similar past behavior as the current user; (ii) *Content-based filtering* methods use content attributes of the liked/bought items to recommend similar items to the users; (iii) *Hybrid* recommender systems combine collaborative and content-based filtering methods to achieve better performance and circumvent the drawbacks of the two when used separately.

Query Log Mining Query Log Mining (QLM) is a special type of Web usage mining [17], focused on mining log files from Web search engines. QLM techniques are used in variety of applications [171, 172], predominantly being used to improve the ranking and the runtime efficiency of search engines. For example, QLM

approaches can be used to discover patterns and knowledge that can be used for future query refinement and expansion [173], personalized query recommendation and suggestion [174, 175], resolving ambiguity and intent [176], and removing spam. Another important application of QLM is in *search advertising*, also known as sponsored search, where QLM methods are being used to deliver the best matching advertisements to the users based on their search queries and preferences, thus maximizing the revenue of the search engines [177, 178]. *Contextual advertising* is another type of advertising, which tries to match ads with the context displayed in the current Web page [177], for which methods from Web content mining and Web usage mining are used. Furthermore, search queries can be seen as signals in a domain over time and can be represented as time-series. Thus, time-series analysis approaches can be applied on query logs to identify new trends, events, interests, and preferences [179], e.g., political events [180] or virus epidemics [181].

6 The Semantic Web and Semantic Web Mining

While the current Web is intended to be human readable, the Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. Semantic Web technologies facilitate building a large-scale Web of machine-readable and machine-understandable knowledge and thus facilitate data reuse and integration. The basics of the Semantic Web were set by Tim Berners-Lee in 2001 [22], which later lead to the creation of the Linked Open Data [182].[2] Linked Open Data (LOD) is an open, interlinked collection of datasets on the Web in machine-interpretable form, covering many domains [183]. Currently, more than 1000 datasets are interlinked in the Linked Open Data cloud.[3] Since the beginning, the Semantic Web has promoted a graph-based representation of knowledge, e.g., using the Resource Description Framework (RDF).[4] In general, RDF is a framework which provides capabilities to describe information about resources. The core structure of RDF is a set of triples, each consisting of a subject, a predicate, and an object, e.g., db:Berlin dbo:capitalOf db:Germany represents a triple. A set of such triples is called an RDF graph. The term used to describe such RDF graphs has been evolving through the years, i.e., in the beginning they were called *Ontologies* [184], while currently are known as *Semantic Web Knowledge Graphs* or simply *Knowledge Graphs* [185].

In the last decade, a vast amount of approaches have been proposed which combine methods from data mining and knowledge discovery with Semantic Web knowledge graphs. The goal of those approaches is to support different data mining tasks, or to improve the Semantic Web itself. All those approaches can be divided

[2] https://www.w3.org/DesignIssues/LinkedData.html.

[3] https://lod-cloud.net/.

[4] https://www.w3.org/RDF/.

into three broader categories [28, 29]: (i) Using machine learning techniques to create and improve Semantic Web data; (ii) Using data mining techniques to mine the Semantic Web, also called Semantic Web Mining; (iii) Using Semantic Web based approaches, Semantic Web Technologies, and Linked Open Data to support the process of knowledge discovery and data mining.

In the very beginning, a lot of effort was put into building the Semantic Web and Semantic Web datasets, which required use of data mining and machine learning, e.g., extracting structured data from text or Web pages, which is also known as ontology learning [23, 24]. Semantic Web mining became very popular, allowing for formal querying and reasoning over ontological knowledge bases [25, 26, 27]. Because of its structured nature, Semantic Web data has been heavily used as a background knowledge in various machine learning tasks and applications [28, 29]. With the rapid development of deep learning the most popular consumption of Semantic Web Knowledge graphs in data mining is by using *Knowledge Graph Embeddings* [186, 187]. Such algorithms embed the entities and relations, in some cases even bigger components, into a continuous vectors space, where each component is represented with an n-dimensional vector while preserving the information and structure from the graph. Such representation allows easy use of the knowledge graph in various tasks and applications [188, 189].

Besides Linked Open Data and Semantic Web Knowledge Graphs, semantic annotations in HTML pages are another realization of the Semantic Web. Semantic annotations are integrated into the code of HTML pages using one of the four markup languages Microformats,[5] RDFa,[6] Microdata[7] and JSON-LD.[8] Such markup languages extend the standard HTML markup with additional set of attributes and can be automatically recognized, e.g., by a machine. Such semantic annotations are used by search engine companies, such as Bing, Google, Yahoo!, and Yandex. They use semantic annotations from crawled Web pages to enrich the presentation of search results and to complement their knowledge bases [190]. There are several initiatives to extract such data from the whole Web and make it publicly available, such as the *Web Data Commons*.[9] Such data has been used in many e-commerce applications, such as product matching and product categorization [191, 192, 193].

[5] http://microformats.org/.

[6] https://www.w3.org/TR/rdfa-core/.

[7] https://www.w3.org/TR/microdata/.

[8] https://www.w3.org/TR/json-ld11/.

[9] http://webdatacommons.org/.

References

1. T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, A. Secret, The world-wide web, Communications of the ACM 37 (8) (1994) 76–82.
2. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery in databases, AI magazine 17 (3) (1996) 37–37.
3. S. Chakrabarti, Mining the Web: Discovering knowledge from hypertext data, Elsevier, 2002.
4. H. Chen, M. Chau, Web mining: Machine learning for web applications, Annual review of information science and technology 38 (1) (2004) 289–329.
5. B. Liu, Web data mining: exploring hyperlinks, contents, and usage data, Springer Science & Business Media, 2011.
6. R. Kosala, H. Blockeel, Web mining research: A survey, ACM SIGKDD Explorations Newsletter 2 (1) (2000) 1–15.
7. Q. Zhang, R. S. Segall, Web mining: a survey of current research, techniques, and software, International Journal of Information Technology & Decision Making 7 (04) (2008) 683–720.
8. B. Singh, H. K. Singh, Web data mining research: a survey, in: 2010 IEEE International Conference on Computational Intelligence and Computing Research, IEEE, 2010, pp. 1–10.
9. K. Sharma, G. Shrivastava, V. Kumar, Web mining: Today and tomorrow, in: 2011 3rd International Conference on Electronics Computer Technology, Vol. 1, IEEE, 2011, pp. 399–403.
10. F. Johnson, S. K. Gupta, Web content mining techniques: a survey, International Journal of Computer Applications 47 (11).
11. C. E. Dinucă, D. Ciobanu, Web content mining, Annals of the University of Petrosani. Economics 12 (2012) 85–92.
12. A. Herrouz, C. Khentout, M. Djoudi, Overview of web content mining tools, arXiv preprint arXiv:1307.1024.
13. M. O. Samuel, A. I. Tolulope, O. O. Oyejoke, A systematic review of current trends in web content mining, in: Journal of Physics: Conference Series, Vol. 1299, IOP Publishing, 2019, p. 012040.
14. J. Fürnkranz, Web structure mining, Exploiting the Graph Structure of the World-Wide Web, Österreichische Gesellschaft für Artificial Intelligence (ÖGAI) (2002) 17–26.
15. P. R. Kumar, A. K. Singh, Web structure mining: exploring hyperlinks and algorithms for information retrieval, American Journal of applied sciences 7 (6) (2010) 840.
16. R. Jain, D. G. Purohit, Page ranking algorithms for web mining, International journal of computer applications 13 (5) (2011) 22–25.
17. J. Srivastava, R. Cooley, M. Deshpande, P.-N. Tan, Web usage mining: Discovery and applications of usage patterns from web data, ACM SIGKDD Explorations Newsletter 1 (2) (2000) 12–23.
18. J. Vellingiri, S. C. Pandian, A survey on web usage mining, Global Journal of Computer Science and Technology.
19. T. Hussain, S. Asghar, N. Masood, Web usage mining: A survey on preprocessing of web log file, in: 2010 International Conference on Information and Emerging Technologies, IEEE, 2010, pp. 1–6.
20. L. Grace, V. Maheswari, D. Nagamalai, Analysis of web logs and web user in web mining, arXiv preprint arXiv:1101.5668.
21. V. Chitraa, D. Davamani, A. Selvdoss, A survey on preprocessing methods for web usage data, arXiv preprint arXiv:1004.1257.
22. T. Berners-Lee, J. Hendler, O. Lassila, et al., The semantic web, Scientific American 284 (5) (2001) 28–37.
23. V. Tresp, M. Bundschus, A. Rettinger, Y. Huang, Towards machine learning on the semantic web, in: Uncertainty reasoning for the Semantic Web I, Springer,
24. A. Rettinger, U. Lösch, V. Tresp, C. d'Amato, N. Fanizzi, Mining the semantic web, Data Mining and Knowledge Discovery 24 (3) (2012) 613–662 2006, pp. 282–314.

25. Q. K. Quboa, M. Saraee, A state-of-the-art survey on semantic web mining, Intelligent Information Management 5 (01) (2013) 10.
26. D. Dou, H. Wang, H. Liu, Semantic data mining: A survey of ontology-based approaches, in: Proceedings of the 2015 IEEE 9th international conference on semantic computing (IEEE ICSC 2015), IEEE, 2015, pp. 244–251.
27. K. Sridevi, D. R. UmaRani, A survey of semantic based solutions to web mining, International Journal of Emerging Trends and Technology in Computer Science (IJETTS) 1.
28. P. Ristoski, H. Paulheim, Semantic web in data mining and knowledge discovery: A comprehensive survey, Web semantics: science, services and agents on the World Wide Web 36 (2016) 1–22.
29. P. Ristoski, Exploiting semantic web knowledge graphs in data mining, Vol. 38, IOS Press, 2019.
30. Wendy Hall and Thanassis Tiropanis. Web evolution and web science. *Computer Networks*, 56(18):3859–3865, 2012.
31. Christopher Olston, Marc Najork, et al. Web crawling. *Foundations and Trends® in Information Retrieval*, 4(3):175–246, 2010.
32. SM Pavalam, SV Kashmir Raja, Felix K Akorli, and M Jawahar. A survey of web crawler algorithms. *International Journal of Computer Science Issues (IJCSI)*, 8(6):309, 2011.
33. Manish Kumar, Rajesh Bhatia, and Dhavleesh Rattan. A survey of web crawlers for information retrieval. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(6):e1218, 2017.
34. Blaž Novak. A survey of focused web crawling algorithms. *Proceedings of SIKDD*, 5558:55–58, 2004.
35. Yong-Bin Yu, Shi-Lei Huang, Nyima Tashi, Huan Zhang, Fei Lei, and Lin-Yang Wu. A survey about algorithms utilized by focused web crawler. *Journal of Electronic Science and Technology*, 16(2):129–138, 2018.
36. Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer networks*, 33(1-6):309–320, 2000.
37. Debora Donato, Stefano Leonardi, Stefano Millozzi, and Panayiotis Tsaparas. Mining the inner structure of the web graph. In *WebDB*, pages 145–150. Citeseer, 2005.
38. Jonathan JH Zhu, Tao Meng, Zhengmao Xie, Geng Li, and Xiaoming Li. A teapot graph and its hierarchical structure of the Chinese web. In *Proceedings of the 17th international conference on World Wide Web*, pages 1133–1134, 2008.
39. M Ángeles Serrano, Ana Maguitman, Marián Boguñá, Santo Fortunato, and Alessandro Vespignani. Decoding the structure of the www: A comparative analysis of web crawls. *ACM Transactions on the Web (TWEB)*, 1(2):10–es, 2007.
40. Dimitris Achlioptas, Aaron Clauset, David Kempe, and Cristopher Moore. On the bias of traceroute sampling: or, power-law degree distributions in regular graphs. *Journal of the ACM (JACM)*, 56(4):1–28, 2009.
41. Robert Meusel, Sebastiano Vigna, Oliver Lehmberg, and Christian Bizer. Graph structure in the web—revisited: a trick of the heavy tail. In *Proceedings of the 23rd international conference on World Wide Web*, pages 427–432, 2014.
42. Robert Meusel, Sebastiano Vigna, Oliver Lehmberg, and Christian Bizer. The graph structure in the web–analyzed on different aggregation levels. *The Journal of Web Science*, 1, 2015.
43. Oliver Lehmberg, Robert Meusel, and Christian Bizer. Graph structure in the web: aggregated by pay-level domain. In *Proceedings of the 2014 ACM conference on Web science*, pages 119–128, 2014.
44. R. Feldman, I. Dagan, Knowledge discovery in textual databases (KDT)., in: KDD, Vol. 95, 1995, pp. 112–117.
45. C. C. Aggarwal, C. Zhai, Mining text data, Springer Science & Business Media, 2012.
46. M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, K. Kochut, A brief survey of text mining: Classification, clustering and extraction techniques, arXiv preprint arXiv:1707.02919.

47. S. Büttcher, C. L. Clarke, G. V. Cormack, Information retrieval: Implementing and evaluating search engines, MIT Press, 2016.
48. W. B. Croft, D. Metzler, T. Strohman, Search engines: Information retrieval in practice, Vol. 520, Addison-Wesley Reading, 2010.
49. C. D. Manning, P. Raghavan, H. Schütze, Introduction to information retrieval, Cambridge university press, 2008.
50. G. Miner, J. Elder IV, A. Fast, T. Hill, R. Nisbet, D. Delen, Practical text mining and statistical analysis for non-structured text data applications, Academic Press, 2012.
51. A. K. Uysal, S. Gunal, The impact of preprocessing on text classification, Information Processing & Management 50 (1) (2014) 104–112.
52. R. Baeza-Yates, B. Ribeiro-Neto, et al., Modern information retrieval, Vol. 463, ACM press New York, 1999.
53. J. M. Ponte, W. B. Croft, A language modeling approach to information retrieval, in: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, 1998, pp. 275–281.
54. G. Amati, Information Retrieval Models, Springer New York, New York, NY, 2018, pp. 1976–1981.
55. A. Gani, A. Siddiqa, S. Shamshirband, F. Hanum, A survey on indexing techniques for big data: taxonomy and performance evaluation, Knowledge and information systems 46 (2) (2016) 241–284.
56. S. E. Robertson, Overview of the okapi projects, Journal of documentation.
57. C. Zhai, Statistical language models for information retrieval, Synthesis Lectures on Human Language Technologies 1 (1) (2008) 1–141.
58. T.-Y. Liu, et al., Learning to rank for information retrieval, Foundations and Trends® in Information Retrieval 3 (3) (2009) 225–331.
59. T.-Y. Liu, Learning to Rank for Information Retrieval., Springer, 2011.
60. C. Carpineto, G. Romano, A survey of automatic query expansion in information retrieval, ACM Computing Surveys (CSUR) 44 (1) (2012) 1–50.
61. J. Ooi, X. Ma, H. Qin, S. C. Liew, A survey of query expansion, query suggestion and query refinement techniques, in: 2015 4th International Conference on Software Engineering and Computer Systems (ICSECS), IEEE, 2015, pp. 112–117.
62. H. K. Azad, A. Deepak, Query expansion techniques for information retrieval: A survey, Information Processing & Management 56 (5) (2019) 1698–1735.
63. R. Dale, The return of the chatbots, Natural Language Engineering 22 (5) (2016) 811–817.
64. A. Følstad, P. B. Brandtzæg, Chatbots and the new world of HCI, interactions 24 (4) (2017) 38–42.
65. D. Diefenbach, V. Lopez, K. Singh, P. Maret, Core techniques of question answering systems over knowledge bases: a survey, Knowledge and Information systems 55 (3) (2018) 529–569.
66. S. Vakulenko, Knowledge-based conversational search, arXiv preprint arXiv:1912.06859.
67. I. Russell, Z. Markov, T. Neller, Web document classification, Jun 3 (2005) 1–19.
68. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.
69. J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
70. Y. Kim, Convolutional neural networks for sentence classification, arXiv preprint arXiv:1408.5882.
71. Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE transactions on neural networks 5 (2) (1994) 157–166.
72. S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, et al., Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001).
73. J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805.

74. P. Berkhin, A survey of clustering data mining techniques, in: Grouping multidimensional data, Springer, 2006, pp. 25–71.
75. M. Steyvers, T. Griffiths, Probabilistic topic models, Handbook of latent semantic analysis 427 (7) (2007) 424–440.
76. L. Chiticariu, M. Danilevsky, H. Ho, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, S. Vaithyanathan, H. Zhu, Web information extraction. (2018).
77. D. Nadeau, S. Sekine, A survey of named entity recognition and classification, Lingvisticae Investigationes 30 (1) (2007) 3–26.
78. G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, C. Dyer, Neural architectures for named entity recognition, arXiv preprint arXiv:1603.01360.
79. V. Yadav, S. Bethard, A survey on recent advances in named entity recognition from deep learning models, arXiv preprint arXiv:1910.11470.
80. I. Segura Bedmar, P. Martínez, M. Herrero Zazo, Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (DDIExtraction 2013), Association for Computational Linguistics, 2013.
81. M. Collins, Y. Singer, Unsupervised models for named entity classification, in: 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999.
82. S. Zhang, N. Elhadad, Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts, Journal of biomedical informatics 46 (6) (2013) 1088–1098.
83. G. Zhou, J. Su, Named entity recognition using an hmm-based chunk tagger, in: proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2002, pp. 473–480.
84. S. Liu, B. Tang, Q. Chen, X. Wang, Effects of semantic features on machine learning-based drug name recognition systems: word embeddings vs. manually constructed dictionaries, Information 6 (4) (2015) 848–865.
85. Y. Li, K. Bontcheva, H. Cunningham, SVM based learning system for information extraction, in: International Workshop on Deterministic and Statistical Methods in Machine Learning, Springer, 2004, pp. 319–339.
86. R. Collobert, J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, in: Proceedings of the 25th international conference on Machine learning, 2008, pp. 160–167.
87. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, Journal of machine learning research 12 (Aug) (2011) 2493–2537.
88. Z. Huang, W. Xu, K. Yu, Bidirectional LSTM-CRF models for sequence tagging, arXiv preprint arXiv:1508.01991.
89. Y. Kim, Y. Jernite, D. Sontag, A. M. Rush, Character-aware neural language models, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016.
90. R. C. Bunescu, R. J. Mooney, A shortest path dependency kernel for relation extraction, in: HLT/EMNLP, ACL, 2005, pp. 724–731.
91. A. Culotta, J. Sorensen, Dependency tree kernels for relation extraction, in: ACL, ACL, 2004, p. 423.
92. R. J. Mooney, R. C. Bunescu, Subsequence kernels for relation extraction, in: NIPS, 2006, pp. 171–178.
93. D. Zelenko, C. Aone, A. Richardella, Kernel methods for relation extraction, Journal of machine learning research 3 (2003) 1083–1106.
94. S. Zhao, R. Grishman, Extracting relations with integrated information using kernel methods, in: ACL, ACL, 2005, pp. 419–426.
95. T. H. Nguyen, R. Grishman, Relation extraction: Perspective from convolutional neural networks., in: VS@ HLT-NAACL, 2015, pp. 39–48.
96. D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, et al., Relation classification via convolutional deep neural network, in: COLING, 2014, pp. 2335–2344.

97. N. T. Vu, H. Adel, P. Gupta, et al., Combining recurrent and convolutional neural networks for relation classification, in: NAACL-HLT, 2016, pp. 534–539.
98. I. Augenstein, D. Maynard, F. Ciravegna, Distantly supervised web relation extraction for knowledge base population, Semantic Web 7 (4) (2016) 335–349.
99. A. L. Gentile, Z. Zhang, I. Augenstein, F. Ciravegna, Unsupervised wrapper induction using linked data, in: K-CAP, ACM, 2013, pp. 41–48.
100. G. Ji, K. Liu, S. He, J. Zhao, Distant supervision for relation extraction with sentence-level attention and entity descriptions, in: AAAI, 2017, pp. 3060–3066.
101. A. J. Ratner, C. D. Sa, S. Wu, D. Selsam, C. Ré, Data programming: Creating large training sets, quickly, in: NIPS, 2016, pp. 3567–3575.
102. B. Roth, T. Barth, M. Wiegand, D. Klakow, A survey of noise reduction methods for distant supervision, in: AKBC, ACM, 2013, pp. 73–78.
103. P. Ristoski, A. L. Gentile, A. Alba, D. Gruhl, S. Welch, Large-scale relation extraction from web documents and knowledge graphs with human-in-the-loop, Journal of Web Semantics (2019) 100546.
104. M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni, Open information extraction from the web, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007, pp. 2670–2676.
105. O. Etzioni, A. Fader, J. Christensen, S. Soderland, M. Mausam, Open information extraction: The second generation, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence—Volume One, IJCAI'11, AAAI Press, 2011, pp. 3–10.
106. V. Presutti, A. G. Nuzzolese, S. Consoli, A. Gangemi, D. Reforgiato Recupero, From hyperlinks to semantic web properties using open knowledge extraction, Semantic Web 7 (4) (2016) 351–378.
107. Q. Li, H. Ji, Incremental joint extraction of entity mentions and relations, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2014, pp. 402–412.
108. N. Kushmerick, D. S. Weld, R. Doorenbos, Wrapper induction for information extraction, University of Washington Washington, 1997.
109. N. Dalvi, R. Kumar, M. Soliman, Automatic wrappers for large scale web extraction, Proceedings of the VLDB Endowment 4 (4) (2011) 219–230.
110. M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, Y. Zhang, Webtables: exploring the power of tables on the web, Proceedings of the VLDB Endowment 1 (1) (2008) 538–549.
111. M. Cafarella, A. Halevy, H. Lee, J. Madhavan, C. Yu, D. Z. Wang, E. Wu, Ten years of WebTables, Proceedings of the VLDB Endowment 11 (12) (2018) 2140–2149.
112. G. Limaye, S. Sarawagi, S. Chakrabarti, Annotating and searching web tables using entities, types and relationships, Proceedings of the VLDB Endowment 3 (1-2) (2010) 1338–1347.
113. P. Venetis, A. Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, Recovering semantics of tables on the web.
114. Z. Zhang, Effective and efficient semantic table interpretation using TableMiner+, Semantic Web 8 (6) (2017) 921–957.
115. M. J. Cafarella, A. Halevy, N. Khoussainova, Data integration for the relational web, Proceedings of the VLDB Endowment 2 (1) (2009) 1090–1101.
116. X. Zhang, Y. Chen, J. Chen, X. Du, L. Zou, Mapping entity-attribute web tables to web-scale knowledge bases, in: International Conference on Database Systems for Advanced Applications, Springer, 2013, pp. 108–122.
117. C. S. Bhagavatula, T. Noraset, D. Downey, Methods for exploring and mining tables on wikipedia, in: Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, 2013, pp. 18–26.
118. O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, C. Bizer, The mannheim search join engine, Journal of Web Semantics 35 (2015) 159–166.
119. B. Kruit, P. Boncz, J. Urbani, Extracting novel facts from tables for knowledge graph completion, in: International Semantic Web Conference, Springer, 2019, pp. 364–381.

120. O. Lehmberg, Web table integration and profiling for knowledge base augmentation, Ph.D. thesis (2019).
121. S. Zhang, K. Balog, Web table extraction, retrieval, and augmentation: A survey, ACM Transactions on Intelligent Systems and Technology (TIST) 11 (2) (2020) 1–35.
122. K. McKeown, Text generation, Cambridge University Press, 1992.
123. S. Lu, Y. Zhu, W. Zhang, J. Wang, Y. Yu, Neural text generation: Past, present and beyond, arXiv preprint arXiv:1803.07133.
124. K. Lin, D. Li, X. He, Z. Zhang, M.-t. Sun, Adversarial ranking for language generation, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 3155–3165.
125. Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, L. Carin, Adversarial feature matching for text generation, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 4006–4015.
126. A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, OpenAI Blog 1 (8) (2019) 9.
127. T. Zhang, J. Zhang, C. Huo, W. Ren, Automatic generation of pattern-controlled product description in e-commerce, in: The World Wide Web Conference, 2019, pp. 2355–2365.
128. L. Dong, S. Huang, F. Wei, M. Lapata, M. Zhou, K. Xu, Learning to generate product reviews from attributes, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, 2017, pp. 623–632.
129. J. Ni, J. McAuley, Personalized review generation by expanding phrases and attending on aspect-aware representations, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2018, pp. 706–711.
130. H. Mei, M. Bansal, M. R. Walter, What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment, arXiv preprint arXiv:1509.00838
131. A. Nenkova, K. McKeown, A survey of text summarization techniques, in: Mining text data, Springer, 2012, pp. 43–76.
132. I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.
133. R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, et al., Abstractive text summarization using sequence-to-sequence RNNs and beyond, arXiv preprint arXiv:1602.06023.
134. B. Liu, L. Zhang, A survey of opinion mining and sentiment analysis, in: Mining text data, Springer, 2012, pp. 415–463.
135. K. Ravi, V. Ravi, A survey on opinion mining and sentiment analysis: tasks, approaches and applications, Knowledge-Based Systems 89 (2015) 14–46.
136. L. Zhang, S. Wang, B. Liu, Deep learning for sentiment analysis: A survey, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8 (4) (2018) e1253.
137. R. Zafarani, M. A. Abbasi, H. Liu, Social media mining: an introduction, Cambridge University Press, 2014.
138. S. Vieweg, A. L. Hughes, K. Starbird, L. Palen, Microblogging during two natural hazards events: what twitter may contribute to situational awareness, in: Proceedings of the SIGCHI conference on human factors in computing systems, 2010, pp. 1079–1088.
139. O. Okolloh, Ushahidi, or 'testimony': Web 2.0 tools for crowdsourcing crisis information, Participatory learning and action 59 (1) (2009) 65–70.
140. R. Goolsby, Lifting elephants: Twitter and blogging in global perspective, in: Social computing and behavioral modeling, Springer, 2009, pp. 1–6.
141. A. Schulz, P. Ristoski, H. Paulheim, I see a car crash: Real-time detection of small scale incidents in microblogs, in: Extended semantic web conference, Springer, 2013, pp. 22–33.
142. D. E. O'Leary, Twitter mining for discovery, prediction and causality: Applications and methodologies, Intelligent Systems in Accounting, Finance and Management 22 (3) (2015) 227–247.

143. K. Shu, A. Sliva, S. Wang, J. Tang, H. Liu, Fake news detection on social media: A data mining perspective, ACM SIGKDD Explorations Newsletter 19 (1) (2017) 22–36.
144. S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine.
145. L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: Bringing order to the web., Tech. rep., Stanford InfoLab (1999).
146. J. M. Kleinberg, Authoritative sources in a hyperlinked environment, Journal of the ACM (JACM) 46 (5) (1999) 604–632.
147. S. Wasserman, K. Faust, et al., Social network analysis: Methods and applications, Vol. 8, Cambridge university press, 1994.
148. D. Knoke, S. Yang, Social network analysis, Vol. 154, SAGE Publications, Incorporated, 2019.
149. W. Xing, A. Ghorbani, Weighted PageRank algorithm, in: Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004., IEEE, 2004, pp. 305–314.
150. X. Li, B. Liu, S. Y. Philip, Time sensitive ranking with application to publication search, in: Link Mining: Models, Algorithms, and Applications, Springer, 2010, pp. 187–209.
151. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, Trawling the web for emerging cyber-communities, Computer networks 31 (11-16) (1999) 1481–1493.
152. G. W. Flake, S. Lawrence, C. L. Giles, Efficient identification of web communities, in: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000, pp. 150–160.
153. G. W. Flake, S. Lawrence, C. L. Giles, F. M. Coetzee, Self-organization and identification of web communities, Computer 35 (3) (2002) 66–70.
154. A. Lancichinetti, S. Fortunato, Community detection algorithms: a comparative analysis, Physical review E 80 (5) (2009) 056117.
155. P. Bedi, C. Sharma, Community detection in social networks, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 6 (3) (2016) 115–135.
156. D. Zhang, J. Yin, X. Zhu, C. Zhang, Network representation learning: A survey, IEEE transactions on Big Data.
157. S. Bhagat, G. Cormode, S. Muthukrishnan, Node classification in social networks, in: Social network data analytics, Springer, 2011, pp. 115–148.
158. A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.
159. B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.
160. F. D. Malliaros, M. Vazirgiannis, Clustering and community detection in directed networks: A survey, Physics Reports 533 (4) (2013) 95–142.
161. M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, S. Wang, Learning graph-based poi embedding for location-based recommendation, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 15–24.
162. J. Tang, J. Liu, M. Zhang, Q. Mei, Visualizing large-scale and high-dimensional data, in: Proceedings of the 25th international conference on world wide web, 2016, pp. 287–297.
163. Z. Liu, V. W. Zheng, Z. Zhao, F. Zhu, K. C.-C. Chang, M. Wu, J. Ying, Distance-aware DAG embedding for proximity search on heterogeneous graphs, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
164. J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, arXiv preprint arXiv:1812.08434.
165. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, A comprehensive survey on graph neural networks, arXiv preprint arXiv:1901.00596.
166. J. Wang, Encyclopedia of Data Warehousing and Mining, (4 Volumes), iGi Global, 2009.
167. D. Tanasa, B. Trousse, Advanced data preprocessing for intersites web usage mining, IEEE Intelligent Systems 19 (2) (2004) 59–65.

168. F. Ricci, L. Rokach, B. Shapira, Introduction to recommender systems handbook, in: Recommender systems handbook, Springer, 2011, pp. 1–35.
169. J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowledge-based systems 46 (2013) 109–132.
170. S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Computing Surveys (CSUR) 52 (1) (2019) 1–38.
171. F. Silvestri, et al., Mining query logs: Turning search usage data into knowledge, Foundations and Trends® in Information Retrieval 4 (1–2) (2009) 1–174.
172. A. Al-Hegami, H. Al-Omaisi, Data mining techniques for mining query logs in web search engines.
173. H. Cui, J.-R. Wen, J.-Y. Nie, W.-Y. Ma, Probabilistic query expansion using query logs, in: Proceedings of the 11th international conference on World Wide Web, 2002, pp. 325–332.
174. R. Baeza-Yates, C. Hurtado, M. Mendoza, Query recommendation using query logs in search engines, in: International conference on extending database technology, Springer, 2004, pp. 588–596.
175. M. Speretta, S. Gauch, Personalized search based on user search histories, in: The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), IEEE, 2005, pp. 622–628.
176. B. J. Jansen, D. L. Booth, A. Spink, Determining the user intent of web search engine queries, in: Proceedings of the 16th international conference on World Wide Web, 2007, pp. 1149–1150.
177. K. Dave, V. Varma, et al., Computational advertising: Techniques for targeting relevant ads, Foundations and Trends® in Information Retrieval 8 (4–5) (2014) 263–418.
178. D. Hillard, S. Schroedl, E. Manavoglu, H. Raghavan, C. Leggetter, Improving ad relevance in sponsored search, in: Proceedings of the third ACM international conference on Web search and data mining, 2010, pp. 361–370.
179. M. Vlachos, C. Meek, Z. Vagena, D. Gunopulos, Identifying similarities, periodicities and bursts for online search queries, in: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, 2004, pp. 131–142.
180. I. Weber, V. R. K. Garimella, E. Borra, Mining web query logs to analyze political issues, in: Proceedings of the 4th annual ACM web science conference, 2012, pp. 330–334.
181. P. M. Polgreen, Y. Chen, D. M. Pennock, F. D. Nelson, R. A. Weinstein, Using internet searches for influenza surveillance, Clinical infectious diseases 47 (11) (2008) 1443–1448.
182. C. Bizer, T. Heath, T. Berners-Lee, Linked Data—The Story So Far., Int. J. Semantic Web Inf. Syst. 5 (3) (2009) 1–22.
183. M. Schmachtenberg, C. Bizer, H. Paulheim, Adoption of the linked data best practices in different topical domains, in: International Semantic Web Conference, Springer, 2014, pp. 245–260.
184. S. Staab, R. Studer, Handbook on ontologies, Springer Science & Business Media, 2010.
185. H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, Semantic web 8 (3) (2017) 489–508.
186. Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Transactions on Knowledge and Data Engineering 29 (12) (2017) 2724–2743.
187. H. Cai, V. W. Zheng, K. C.-C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Transactions on Knowledge and Data Engineering 30 (9) (2018) 1616–1637.
188. P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowledge-Based Systems 151 (2018) 78–94.
189. P. Ristoski, J. Rosati, T. Di Noia, R. De Leone, H. Paulheim, RDF2Vec: RDF graph embeddings and their applications, Semantic Web 10 (4) (2019) 721–752.
190. R. Meusel, Web-scale profiling of semantic annotations in html pages, Ph.D. thesis (2017).

191. P. Petrovski, A. Primpeli, R. Meusel, C. Bizer, The WDC gold standards for product feature extraction and product matching, in: International Conference on Electronic Commerce and Web Technologies, Springer, 2016, pp. 73–86.
192. P. Ristoski, P. Petrovski, P. Mika, H. Paulheim, A machine learning approach for product matching and categorization, Semantic web (Preprint) (2018) 1–22.
193. Z. Zhang, M. Paramita, Product classification using microdata annotations, in: International Semantic Web Conference, Springer, 2019, pp. 716–732.

Mining Temporal Data

Robert Moskovitch

1 Introduction

With the introduction of the internet, an exponential growth in text based data, which required new methods to analyze text, retrieve, understand, classify, and more. However, with the recent growing adoption of the Internet of Things, in which devices in various domains become connected to the internet network and produce multitude of logged longitudinal data, we are facing a similar increase in multivariate temporal data that is generated by various devices and sensors. Such data can be logged from smart watches, cars, smart refrigerators, smart phones, televisions, sensors in the space of rooms for various purposes, and many more, which can be useful in various domains, and for different tasks. This trend is expected to increase significantly and requires new methods and more scalability. An extension of temporal data analytics, which are out of the scope of this paper, is spatio-temporal data analysis (Gong et al., 2020a,b; Wang et al., 2019; Zhang et al., 2018; Gong et al., 2018), which introduces more challenges.

New temporal data may come from specific mobile apps, such as care home careers that are documenting activities related to the residents, and more. For various tasks, such as monitoring residents in care homes, for example, it is possible to combine data coming from various sources, such as the space sensors in the room,

The authors wish to thank the Israeli Ministry of Science and Technology, who assisted in funding this project with grants 8760441 and 8760521.

R. Moskovitch (✉)
Software and Information Systems Engineering, Ben Gurion University, Beer Sheva, Israel

Population Health Science and Policy, Icahn School of Medicine at Mount Sinai, New York, NY, USA
e-mail: robertmo@bgu.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_21

or sensors on their bed, smart watches, the care home activities log, and more. For example, in order to build a prediction model for residents' falls in care homes (Dvir et al., 2020), one would like to have data from various sources, the resident's personal phone, or perhaps smart watch, or some monitoring bracelet, data from an app that collects data from the caregivers, as well as data from their clinical database, and more devices in their room if available. Having such a variety of data stores many challenges but also opportunities for contributions in many domains.

However, various sources of temporal data result often with heterogeneous sampling forms along time, as shown in Fig. 1. Incorporating various temporal variables, having different forms of temporal measurements is one of the most challenging topics in temporal data analysis. Figure 1 presents five forms of temporal variables (a) is a variable that is measured in a fixed frequency, often comes from electronic sensors' measurements, or a routine measurement. For example, a patient blood pressure monitored in an Intensive Care Unit; (b) is also a fixed frequency variable but is measured in a different frequency—a common situation, especially when gathering data from various sources, and the variables may be sampled in a different frequency;

(c) is a time point values series irregularly sampled, having varying durations between the measurements, often coming from manual measurements, or based on an event driven. For example, each time a patient visits the clinic; (d) is a sequence of symbolic events that have no duration, or having a single time unit duration, which may have various symbols and varying durations among the events, and multiple symbolic events can happen in the same timestamp. The symbols can be, for example, multiple products that were purchased in a single purchase at the supermarket; (e) is a sequence of symbolic time intervals series, which represents a series of symbolic events that have varying durations.

Symbolic time intervals can be raw, or the result of abstracting time series (Shahar, 1997; Höppner, 2002). An intuitive example for raw symbolic time intervals is the duration of traffic lights in a roads conjunction, whether when it is red, or green, or yellow. Other examples from the medical domain may be drug exposers—a period of time a patient is prescribed for a drug, or patients' condition and cet. There are methods to analyze part of the variables' types that were shown in Fig. 1, but no methods that incorporate all the types together naturally without any type of transformation. For example, most of the methods for time series analysis, whether for forecasting, or when using Hidden Markov Models, or when using deep learning methods, such as recurrent neural networks, or convolutional neural networks, expect all the variables to be sampled in the same frequency, such as series a or b in Fig. 1. In the description of Fig. 1, we will refer to examples from the medical domain and will follow up with these examples along the paper. Thus, the series a and b in Fig. 1 can refer to sensory data that comes from ICU, or from a smart watch or a person.

Irregularly sampled variables, such as blood tests that are measured every once and a while, bring even more challenges, and one way to overcome these challenges is the use of imputation (Pratama et al., 2016), which for irregularly sampled variables can be limited, especially when the data is severely sparse. For

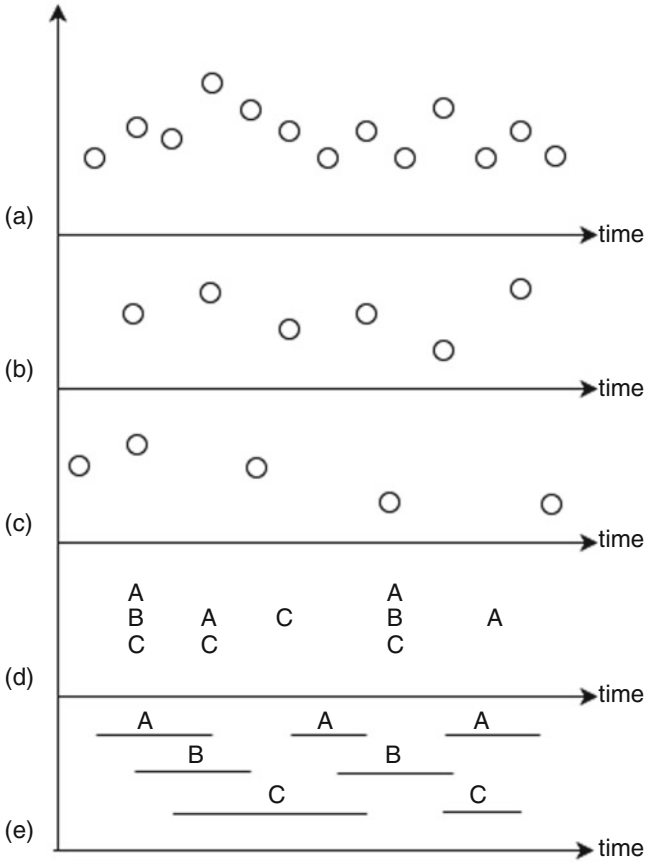


Fig. 1 Temporal data datasets often include variables sampled and represented in various forms. Incorporating all the variables within the same analysis method is one of the challenges in temporal data analytics

sequential data, such as series c, which may describe conditions or procedures, there are methods that are known as sequential mining that discover frequent sequences of symbolic events, which are discussed in more details in the subsection about sequential mining. These frequent patterns can be used later for knowledge discovery, for sequence prediction (Gueniche et al., 2013) to predict the next item, or as features for classification (Moskovitch et al., 2015). Lastly for symbolic time intervals, such as series d in Fig. 1, which may represent drug exposers for example, similar to sequential patterns discovery, there are time intervals mining methods that can discover frequent symbolic time intervals related patterns (Papapetrou et al., 2009; Moskovitch et al., 2015; Harel and Moskovitch, 2021). However, in order to incorporate all the variables into the same analysis method transformations have to be made, as will be discussed later in the section about temporal abstraction.

In this chapter the intention is to cover in high level the field of mining temporal data, which becomes increasingly more important in various domains, due to the access to data, and potential implications, given the space limitations. We start with going through the various methods available for each type of temporal data and discuss challenges and needs.

Then, we focus more on methods that are related more to "mining" patterns in temporal data, which are from the family of patterns' discovery, whether sequential or time intervals patterns mining. Then, we discuss their potential use in various tasks, such as classification. Finally, we discuss the domain, its challenges and open problems, which may lead to potential opportunities.

2 Time Point Series Mining

When referring to series a, b, or c in Fig. 1, they are called often time point series, representing variables whose values were measured in specific time points. Such data can be univariate, which refers to a single time series, as happens with stocks data, for example, or multivariate, which refers to a collection of time series variables (for example, when measuring a human, it will be a collection of the heart rate, blood pressure, glucose levels time series, which are multivariate, unlike only heartrate, which is univariate in case it is alone). In stocks data, for example, adding to a specific stock that we would like to forecast, other stocks data that are expected to be predictive (stocks from the same sector, or stocks that correlate), or other relevant longitudinal data, such as coin values, economical measures and more, which will result in a multivariate forecasting problem.

Often when referring to such data, it is referred to as time series analysis, which by that people often refer to forecasting, or other methods that often expect the time series to be regularly measured, or classification, in which the focus is on the features that can be extracted for the classification model, or clustering of time series, which is a somewhat limited research area (Schulam et al., 2015) which has room for contributions, as will be discussed. Thus, there is a value for each time point, whether it is univariate or multivariate. Here is a summary of such relevant methods, which were developed in various fields, such as statistics, data mining, and generally mathematics.

One of the popular, or useful, methods in time series analysis is forecasting, in which there is an intention to forecast future values of a variable, typically based on its earlier values, or in addition based on more variables. Such need has many implications in real life problems, from forecasting stocks, inventory management, various business events and decisions, through weather forecast, medical conditions, and more. However, in order to be able to forecast values of a time series, based on its past values, the time series has to fulfil the stationary requirements.

That means that the time series is generated by a process that does not change along time. For that there are tests, such as dual Kalman filter that enables after several differencing to stationarize the data. It is crucial to stationarize the data,

otherwise it is impossible to learn a forecasting model, since the variable changes their "behavior" along time, which means it is not predictable. For forecasting there are two main basic techniques: Autoregression and Moving Average, as well as deep learning based. Autoregression is based on regression models, in which the variable forecasting is modelled based on its values in previous time stamps. We would like to learn a regression model that describes the relations between the variable's recent values, given the time axis, which is an autoregression. Moving Average is a simple forecasting method that is based on the average of the last k values. A more advanced way is weighted moving average that includes weights for each of the k time points values, but it is a challenge to determine the weights, which is often done through minimizing the models' mean error. Combining these two approaches was made by (Moran and Whittle, 1951) who brought the Autoregression and Moving Average (ARMA) model, which enables better forecasting. However, ARMA may be vulnerable to non-stationary time series. For that an advanced version called Autoregressive Integrated Moving Average (ARIMA) was developed (Box and Jenkins, 1970; Hamilton, 1994) which includes differencing to stationarize the time series. ARMA includes two parameters, p—that determines the number of earlier time point values relatively to the predicted value at time t, and q—that determines the number of earlier errors, based on which the predicted value is forecasted. The i parameter determines the number of differencing that is performed to stationarize the time series. The challenge in ARIMA is determining the p, i, q parameters' values, and for that there are algorithms and packages. Recently, some packages were developed in the industry that make the use of these ideas simpler and more automated (Taylor and Letham, 2017). Although when referring to forecasting it is often univariate, there are extensions that enable to employ additional variables to perform forecasting based on multivariate time series.

3 Classification with Time Series

However, except forecasting there are other tasks often used that consist of fixed frequency sampled temporal variables (Fig. 1a,b), such as classification which is probably the most important tasks with the growth in temporal data availability.

Classification of multivariate time series is useful to classify but also to can be used to predict based on a sliding window (Schvetz et al., 2021; Itzhak et al., 2020; Novitski et al., 2020). However, in classification the main focus and challenge is in the extraction of features from the multivariate temporal data, which becomes even more complex with heterogeneous temporal variables, as will be discussed in later sections.

In a recent comprehensive study on time series classification (Bagnall et al., 2016) the authors list the following options for the task of multivariate time series classification: using the whole series, in which the values are used as a features vector which is fed to a classifier, or similarity measures (Höppner, 2016) that are used to compare the time series' values vector to another vector of time series,

or a centroid of a class, in which most research had focused on having elastic similarity measures (Ratanamahatana and Keogh, 2005; Rakthanmanon et al., 2013; Lines and Bagnall, 2014). Intervals, which refers to subsets of the time series, in which the best (given some criteria) "interval" may be selected, can be useful as features for classification, although using multiple "intervals" is favorable (Lines and Bagnall, 2014; Diez et al., 2005). Shapelets are a good example of such intervals, which are short subsets of time series that enables to classify the time series in a database, based on the similarity of the time series to a Shapelet (Ye and Keogh, 2010; Rakthanmanon et al., 2013). Dictionary based, in which the frequency of subseries is used to represent the time series; Combinations, which refer to combining several approaches for the representation of the time series; and Model based, which include, Hidden Markov Models (Kotsifakos and Papapetrou, 2014), extracting various features for classification, such as Fourier transform, using various similarity measures beyond the Euclidean, such as the Dynamic Time Warping (DTW), or Shapelets and other. As well as of course with the recent increasing use and employment of neural networks methods (LeCun et al., 2015), such as Recurrent Neural Networks (RNN) (Hochreiter and Schmidhuber, 1997) or as they are often mostly used as Long Short Term Memory (LSTM) networks and Convolutional Neural Networks (CNN) (Krizhevsky et al., 2012), on which we elaborate later.

However, before going over neural networks based methods, or in their new branding deep learning, we will go over the various methods that were developed in the past decades mainly for the purpose of classification. A popular approach uses a nearest neighbor classifier, which requires a similarity function to represent the proximity between the various time series, as was compared in (Lines and Bagnall, 2014) using various similarity measures, in which it was found that DTW is the most accurate. Additionally, they had shown that ensembelling NN classifiers, having different similarity measures, outperformed each of the ensemble members. Further improvements to this approach were made by using other types of classifiers (Baydogan et al., 2013; Deng et al., 2013; Bagnall et al., 2016; Bostrom and Bagnall, 2015) which were found to be more effective than NN-DTW (Bagnall et al., 2016).

Nevertheless, Fawaz et al. (2019) criticized the community for not using deep neural networks in comparison studies of ensemble methods (Neamtu et al., 2018; Bagnall et al., 2016; Lines and Bagnall, 2014). The renaissance of neural networks in recent years through the employment of GPUs resulting in Deep Neural Networks (DNN), or deep learning (LeCun et al., 2015), had made significant improvements in classification in some applications, such as image processing. Deep Convolutional Networks had revolutionized the field of computer vision (Krizhevsky et al., 2012). Following the success of DNNs in computer vision, large amount of efforts in proposing various DNN architectures to solve problems in different domains such as Natural Language Processing (NLP) tasks, machine translation, learning word embeddings (Mikolov et al., 2013) document classification (Le and Mikolov, 2014; Goldberg, 2016), and speech recognition (Hinton et al., 2012; Sainath et al., 2013) were made.

The last tasks while not temporal have a sequential nature, which is something DNNs can be effective as well, as will see soon. The main advantage of advanced DNN architectures, such as RNNs and CNNs, is that they "extract" the features through the learning, and there is no need to provide it the features, but this also makes the model hard to be explained and interpreted. The most traditional architecture for deep learning is a multi-layer perceptron, or a fully connected network, in which there are several layers of neurons connected through the layers which are modelled by the weights of the neural network. Typically each value in the data will be entered as a "feature" input to the network. While such architectures can be very effective for classification, in which features that were extracted from the data are given as input, for time series, even univariate, they do not capture the time, since when giving the values of each time stamp as an input the assumption is that they are independent.

Moreover, such network cannot handle time series having varying lengths. For that more advanced architectures were developed or adopted from other tasks to use in time series data. Recurrent Neural Networks were designed mainly to predict are traditionally applied for forecasting, but their use for classification of time series is applied too. Motivated by the successful application of deep Convolutional Neural Networks in computer vision and image processing (LeCun et al., 2015), researchers have been using them recently for classification of time series (Gamboa, 2017). The idea is that multivariate time series, especially fixed frequency sampled, look like a two dimensional image. However, unlike in images, the convolution is applied only in one dimension (of the time) rather than two dimensions. Convolution can be seen as applying a sled filter over the time series. In fact, applying a convolution is like applying a moving average with a sliding window of some length.

Recently, several studies were published showing the successful use of CNN in various domains, for example, in Medicine (Yang et al., 2020; Novitski et al., 2020). Echo State Networks Jaeger and Haas (2004); Gallicchio and Micheli (2017) are DNN based architectures that were designed to overcome the limitation of the use of RNNs (were designed to predict the value of each time stamp in the time series, suffer from the vanishing gradient problem especially on long time series, and hard to parallelize) by avoiding the need to compute the gradient of the hidden layers. Generative models, in which unsupervised methods, are applied to extract features that are later used as features to train a classifier, which are called also model based classifiers. These model based classifiers (Bagnall et al., 2016) include often auto-regressive models (Bagnall and Janacek, 2014), hidden Markov models (Kotsifakos and Papapetrou, 2014), kernel methods (Chen et al., 2013), sequential patterns (Fradkin and Mörchen, 2014), time intervals patterns based Moskovitch et al. (2015); Moskovitch and Shahar (2015); Batal et al. (2012), and more Längkvist et al. (2014).

The contribution of the deep learning space for the generative model includes applying unsupervised learning, such as denoising auto-encoders (Bengio et al., 2013; Hu et al., 2016). A generative CNN-based model was proposed by Wang et al. (2017), Mittelman Mittelman (2015) introducing a deconvolutional operation to reconstruct a multivariate time series. Deep Belief Networks (DBNs) were also

used to model the latent features in an unsupervised manner which were later used to classify univariate and multivariate time series Wang et al. (2017); Banerjee et al. (2017). Self-predict modelling for time series classification using ESNs to reconstruct the time series first, and then the learned representation was utilized for classification (Aswolinskiy et al., 2017; Bianchi et al., 2021; Chouikhi et al., 2018; Ma et al., 2016; Chen et al., 2015, 2013; Che, 2017). A recent comprehensive survey on the use of deep learning for applications in human motion detection based on mobile or sensor networks Nweke et al. (2018) and generally more on the use of deep learning for time series classification can be found in Fawaz et al. (2019).

4 From Time Point Series to Symbolic Temporal Data

However, real life observational datasets typically contain heterogeneous multivariate temporal data, as shown in Fig. 1. Most methods cannot incorporate all these types of data, and one way to do that is by transforming the time point values series is by abstraction. Increasingly it can be seen that more studies transform time point series into a symbolic representation, through a process known as discretization, or temporal abstraction.

Discretization is often used in machine learning, such as in classifiers as decision trees [Quinlan, 1986] and more, to enable generalization. Using discretization transforms continuous variables into a finite symbolic representation, which reduces the values spectrum, and increases generalization. Generalization is good for model learning and for frequent patterns discovery. However, often before abstracting the temporal data, there is a need to decrease the granularity (sometimes called dimensionality (Lin et al., 2003)) of the temporal data, for which Piecewise Aggregate Approximation (PAA) Keogh et al. (2001) can be used. When using PAA it is required to determine a time window, for which a value will be extracted, such as the mean value, and accordingly the resulted granularity. For each time window a representative single value will be calculated, based on its mean value. Note that in many cases the number of values within the time window may vary. For example, with spikes data in multielectrode, or due to missing measurements. Thus, in addition, to the mean value, which is part of the PAA method, other metrics can be extracted, such as the min value, or max, or the number of values measured, median and more. The process of transforming the time point value series data into a symbolic representation is becoming increasingly popular with the development of various discretization methods (Lin et al., 2003; Moskovitch and Shahar, 2015; Ramírez-Gallego et al., 2016) which results in symbolic time series (Fig. 2).

The next step, in which adjacent symbols, having the same symbol, are concatenated into symbolic time intervals (shown at the top of Fig. 1), is called typically Temporal Abstraction Shahar (1997); Höppner (2002). Although Temporal Abstraction often refers to transformation into symbolic time intervals, we will refer to it also when transforming into symbolic time series. Figure 2 shows an illustration of the temporal abstraction process on a raw time point value series

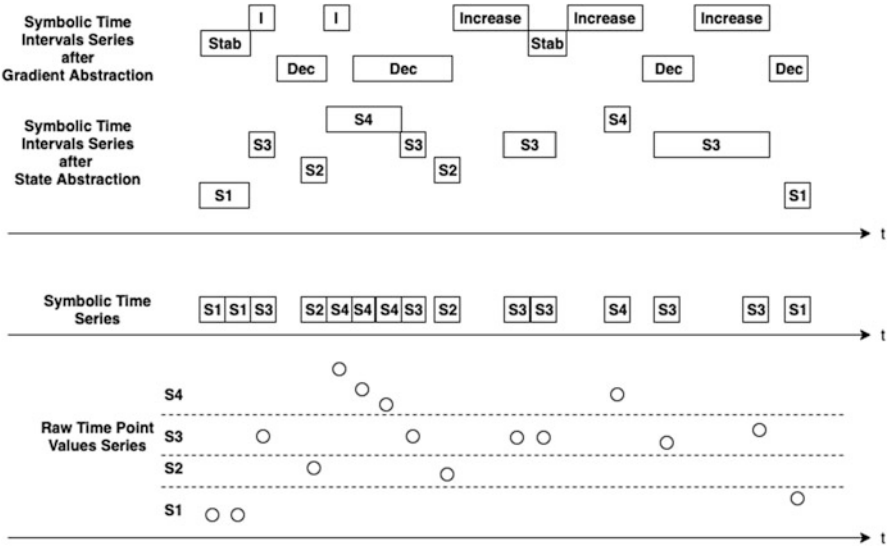


Fig. 2 Symbolic Temporal Data. The time point series at the bottom is transformed into a symbolic time series, based on the cutoffs. Above its transformation into symbolic time intervals series is shown, and at the top its representation after Gradient abstraction

shown in the bottom. The dashed lines represent cutoffs, based on which the values will be discretized into a symbolic representation. These cutoffs can come from the domain, when available, such as cutoffs for blood test that determine whether the value is in normal, or it is high, or low. When the cutoffs come from the domain, it is called Knowledge Based Shahar (1997), alternatively there are various data driven discretization methods that can be used, as mentioned in more details in the next paragraph. In the example in Fig. 2 there are three cutoffs which result in four symbols, or states. At the above illustration the sequence of symbolic time series, in which each value was classified into the appropriate symbol and above each time point value there is a symbol. As mentioned earlier, the cutoffs can be acquired from data driven methods.

There are quite few types of data driven discretization methods (Ramírez-Gallego et al., 2016), but only few were designed specifically for temporal data. Simple and popular discretization methods are Equal Width Discretization, in which the range of values are divided into equal width bin ranges, which results in uneven distribution of the values within the bins. Another simple method is Equal Frequency Discretization, which results in evenly distributed values in the bins, but the ranges width of each bin is different. Among the temporal methods, the Symbolic Aggregate approXimation (SAX) is probably the most popular in recent years (Lin et al., 2003). SAX includes in its first step the application of PAA, and based on the mean and standard deviation of the time series values, states, or symbols, are created, which are derived from the gaussian distribution of the values.

SAX has later been enhanced to include improvements and was used successfully in several tasks (Ueno et al., 2006; Camerra et al., 2010) and applications (Lin et al., 2003).

Other discretization methods that were designed for time series are Persist (Mörchen and Ultsch, 2005) and TD4C (Moskovitch and Shahar, 2015). Persist was developed to enable abstraction that results with the longest time intervals (in order to later discover patterns of coinciding time intervals using the TSKM method Mörchen (2006), on which we will elaborate in the section about time intervals mining). Persist divides the values range into percentiles and having a Kulback–Leibler (Kullback and Leibler, 1951) based measure selects the best cutoffs in a greedy fashion, when the self-transition probabilities of the symbols are larger than the marginal transition probabilities, which guarantees the longest resulted time intervals. However, while all the previous methods are unsupervised, the Temporal Discretization for Classification (TD4C) (Moskovitch and Shahar, 2015) is a supervised method that intends to find cutoffs that result in different distribution of the states in the different classes. Thus, ideally most of the values for each class will be frequent in different states. For that three measures were used Entropy, Cosine Similarity, and Kulback–Leibler (Kullback and Leibler, 1951), in which the Cosine performed best.

Other relevant discretization methods use K-means, and more, which are reported in Ramírez-Gallego et al. (2016). Whatever source of cutoffs were chosen, the time point values are classified into the appropriate state and a symbolic time series are created, as shown in Fig. 2 above the raw time series. As can be seen for each time point value there is a symbol among the four states S1, S2, S3, and S4. Another way to use the discretized values is by abstracting them temporally, which results in symbolic time series, as shown above. At the top, there are two types of Temporal Abstraction. The first is State Abstraction, and the second which is above is Gradient Abstraction.

State Abstraction uses the cutoffs, as explained earlier, but here when the same states appear adjacently they are concatenated into symbolic time intervals. Moreover, when the same symbols have a gap between them (no measurements) then it is possible to define an interpolation function Shahar (1997), which means defines what is the largest duration between two states having the same value (symbol) that we can assume that had the same state during the entire period—as can be seen in the last S3 time interval on the right, which is a sort of knowledge based imputation. Gradient abstraction classifies the time point values according to the first derivative of the last k values series, or of the values in some recent period of time. For that several thresholds can be defined, for example, larger than 0, or 0 or smaller than zero. Obviously, it is better to have instead of 0, some values above or below, so, for example, 0 -/+. Additionally, it can be more than three levels of gradient abstraction.

Similar to discretization and state abstraction, also here we can have for each time point value its corresponding gradient, which will result in symbolic time series. Alternatively, if adjacent symbolic time series having the same gradient value are concatenated, we will get symbolic time intervals series, as shown at the top of

Fig. 2. After explaining how symbolic temporal time series or time intervals data can be created, such data can be then analyzed and used for various purposes, such as discovery of frequent temporal patterns, which can be further used for several purposes, such as temporal knowledge discovery, classification and prediction, or clustering, and more.

5 Mining Sequential Data

Pattern mining is a popular field within data mining, and to some extent perhaps identifies it and is more unique to data mining, rather than other methods that can be identified also with machine learning, or generally artificial intelligence. Popular methods, such as itemset mining and association rules mining, are useful in various domains and put the foundations of this field, although they are not temporal. Typically, the task is to discover items that co-occur, or appear together, in basket bags, or other collections (Zaki, 2000; Pei et al., 2001; Uno et al., 2004). Agrawal and R. Srikant (1994) introduced the AprioriAll algorithm and principle, which still holds and useful in sequential or time intervals mining, defining that a pattern can be frequent only when its components are initially frequent. Thus, in order to find, for example, that the pattern "ab" is frequent, first we have to verify that "a" is frequent and "b" is frequent, otherwise there is no chance that "ab" can be frequent.

Sequential pattern mining to some extent extends itemset mining, in which a sequence may represent a multiple itemsets along time (i.e., instead of a single basket of items in a record, there are multiple baskets of items ordered sequentially in a record along time). Sequential mining (Fournier-Viger et al., 2014) is a private case of time intervals mining (which will be discussed in detail in the next section), in which the events have no duration, or a single time unit duration, which results in a simpler need for temporal relations, including only before (and equal to some extent, as will be explained later). Nevertheless, sequential mining will be discussed before time intervals mining, since it was developed earlier and it is simpler. Sequential data mining is a practical method that is relevant in many potential applications, such as click stream analysis, network data, basket analysis, and more.

It assumes there is a database of sequences of multiple items, such as "abg(ab)k(lm)d." In that example, the items that appear in brackets occur in the same time stamp, thus, (ab) occur together after g and before k, and also (lm) occur together, but abg at the beginning do not occur in the same time, but a happens and then b, followed by g. Thus, the order of the items is meaningful, and each letter, or item, is happening in a separate timestamp, while those in the brackets are multiple items happening in the same time. Areal life example can be a person who goes and buys at the supermarket: first time buys a, then b, then g, and then buys a and b together, later buys k, then l and m together, and then buys d.

Given a database of such sequences, a pattern often will be defined by its frequency in the database—based on the number of sequences that contain it, a metric called support. When referring to the percentage of the records containing

the sequential pattern in the entire sequences database, it is called relative support. The goal of Sequential Pattern Mining is to discover frequent subsequences, ideally interesting, in a database. A subsequence s is frequent, if and only if, support(s) $\geqq$ min_support, for a given minimal support threshold which is determined by the user. Having even a small number of types of items results with a large number of potential combinations of sequences. For that most of the algorithms focus on making the enumeration process and checking whether the patterns are frequent as efficient as possible. Quite few algorithms were proposed for sequential pattern discovery in the past two decades. Some of the most popular are GSP (Srikant and Agrawal, 1996), Spade (Zaki, 2004), PrefixSpan Pei et al. (2004), Spam Ayres et al. (2002), Lapin (Yang and Kitsuregawa, 2005), CM-Spam Fournier-Viger et al. (2014), CM-Spade Fournier-Viger et al. (2014), and more Fournier-Viger et al. (2017); Gan et al. (2019).

It is important to highlight that all the algorithms have the same output of frequent sequential patterns, given a minimal support threshold, but they differ in the way they perform the discovery and the corresponding memory and computation consumption. In general, there are two approaches for mining sequential patterns, the first scans the data ideally once, or more, and then using a candidate generation strategy the patterns are extended and discovered when above the minimal support threshold. Another approach is called often project-based, in which initially the single symbol patterns are discovered, and for each pattern a projected database is created, which is further mined for extended patterns. Additionally, some perform the patterns discovery in a breadth-first search approach and other in a depth-first search approach.

For example, GSP (Srikant and Agrawal, 1996) first scans the entire database for one-sized sequences, then 2-sized sequences are generated, followed by 3-sized sequences and so on until no candidates can be generated due to lack of support. Thus, for example, having a database with 3 symbols A, B, C that are frequent, we will have to generate all the potential candidates containing two symbols (2-sized sequences), including: $< aa >, < ab >, < ac >, < ba >, < bb >, < bc >$ $, < (aa) >, < (ab) >, < (ac) >, < (ba) >, < (bb) >, < (bc) >, < (ca) >, <$ $(cb) >, < (cc) >$. Note that $< ab >$ and $< ba >$ are different, since they represent a different order of these items, but $< (ab) >$ and $< (ba) >$ are the same, since within the brackets the items happen in the same location in the sequence. For that $< (ba) >$ is cancelled. Meanwhile, typically $< (aa) >, < (bb) >, and < (cc) >$ are cancelled, since it is like $< a >$, unless the problem is defined in that way that such sequences can appear. Later those that are frequent are extended for additional symbol and so on, until there is not sufficient support.

Other algorithms, such as Spade (Zaki, 2000), or other (Pei et al., 2004; Ayres et al., 2002; Yang and Kitsuregawa, 2005; Fournier-Viger et al., 2014), perform depth-first search, starting with the one-sized sequences, and proceed with their extended larger sequence, when they are frequent. Often the number of sequences combination is very large, and for that the use of the AprioriAll principle is very effective to prune candidates, whose components are not frequent. A detailed discussion on the sequential patterns discovery algorithms can be found

elsewhere (Fournier-Viger et al., 2017). Additional types and related definitions for sequential patterns mining include Closed Sequential Patterns and Maximal Sequential Patterns. Closed sequential patterns are patterns, who do not have super patterns (extended patterns of them) that have the same support.

Some algorithms discover closed patterns, since they include the information also for the sub-patterns of a closed patterns and that way the output is smaller, since there is no need to print the sub-pattern's information. Maximal sequential patterns are those that are not included in longer patterns. Since often a sequential patterns discovery process can be long and have many discovered patterns, it can be performed with constraints to limit the output only for patterns that correspond to the user constraints' criteria. Several constrains were proposed, such as having a minimum and maximum time between events (Yun and Leggett, 2006; Fournier-Viger et al., 2008). Pei et al. (2006) investigated the incorporation of items constraints, which means defining which items should or should not be included in the pattern, as well as minimum/maximum number of items per pattern which defines the length of the pattern.

They also proposed aggregated constraints on process of items in sequences, based on their average, minimum, maximum, sum, and standard deviation of prices. Except long runtime, and many discovered patterns, another typical challenge with sequential patterns is that they discover only "positive" sequences—the sequences of items that exist, but we do not know about those that do not exist, or at least not for sure. Thus, a negative sequential pattern is a pattern containing the negation of at least one item (Zheng et al., 2009; Dong et al., 2018; Cao et al., 2016; Hsueh et al., 2008; Wang, 2019). Mining negative sequential pattern is more challenging, since it includes more potential candidates and the search space becomes larger. However, one of the most challenging topics of sequential mining, and generally in patterns mining, is the ability to choose the most important patterns. For that various strategies were proposed to favor specific frequent sequences, such as weighting the items (Chang, 2011; Ren et al., 2008; Yun and Leggett, 2006), or including the quantities of the items in their utility (Ahmed et al., 2010; Lan et al., 2014; Alkan and Karagoz, 2015). Other measures were proposed in order to favor frequent sequences, when are used as features for classification (Fradkin and Mörchen, 2014) on which we will elaborate in the section about temporal patterns based classification.

6 Mining Time Intervals Data

Mining symbolic time intervals are a relatively young research field. It was mostly sprung during the past two decades, starting with databases of raw symbolic time intervals, and later with the use of abstracted time series. Earlier methods looked for simple relations, such as containment (VillafaneRoy et al., 2000), but most of the methods use some subset of Allen's temporal relations Allen (1983). Allen has proposed seven temporal relations: before, meet, overlap, starts, finished by,

and equal, and their inverse (for example, B after A, instead of A before B). As was criticized by Mörchen (2006) these suffer from "crispiness" and being sometimes over-detailed for knowledge discovery purposes, however, so far, most of the methods use Allen's relations. For that a more flexible version was proposed (Papapetrou et al., 2009; Moskovitch and Shahar, 2015). One of the earliest studies in the area is that of Villafane et al. (2000), which searches for containments of time intervals in multi-symbolic time interval series. Kam and Fu (2000) were the first to use all of Allen's temporal relations to compose frequent time interval sequences, which they called A1. A1 represented the temporal relation from the current sequence, or pattern, to the next symbolic time interval. However, such representation was ambiguous, since the temporal relations were defined only among the pairs of successive intervals.

That had left the temporal relations among the time intervals ambiguous which could fit a disjunction of temporal relations (shan Kam and Fu, 2000; Moskovitch and Shahar, 2015). Höppner (2001) was the first to define a non-ambiguous representation of time intervals patterns that are based on Allen's relations, by a k^2 matrix, to represent all of the pair-wise relations within a k sized Time Intervals Related Pattern (TIRP). While Hoppner had used a full matrix to represent the temporal relations, which was redundant, since it used both Allen's relations and their inverse, the following methods used only a half matrix. Thus, a TIRP was defined by the sequence of the symbolic time intervals and the conjunction of the temporal relations among each pair of time intervals.

Papapetrou et al. (2009) proposed a hybrid approach H-DFS, which combines first indexing the pairs of the symbolic time intervals and then mining the extended TIRPs in a candidate generation fashion. Papapetrou et al. used only five temporal relations: meets, matches (equal, in terms of Allen's relations), overlaps, contains, and follows, similar to Allen's temporal relations, and introduced an epsilon threshold, to make the temporal relations more flexible.

ARMADA, by Winarko and Roddick (2007), is a projection-based time intervals mining algorithm that uses a candidate generation and a mining iterative approach. In each time a pointer projected database is created, but the algorithm runs through the entire records each time till the end, which can make it slow with long records databases. Wu et al. (2007) proposed TPrefixSpan, which is a modification of the PrefixSpan sequential mining algorithm (Pei et al., 2001), for mining non-ambiguous temporal patterns from time interval based events. They refer to the time intervals as a sequence of state-time and end-time time points and implement a sequential mining approach. Patel et al. (2008) introduced IEMiner—a method inspired by Papapetrou's method, which extends the patterns directly. Patel et al. (2008) had compared their method runtime to TPrefixSpan yi Wu and Chen (2007) and H-DFS (Papapetrou et al., 2009) and found their method to be faster.

Moskovitch et al. introduced the KarmaLego algorithm (Moskovitch et al., 2015; Moskovitch and Shahar, 2015), which extends TIRPs directly, having a data structure that generates candidates each time with a symbolic time interval and a temporal relation, and exploits the transitivity property to generate candidates efficiently. Using the transitivity of the temporal relations enables an efficient

candidate's generation, to avoid naïve candidate generation of candidates that cannot exist in reality and contradict the relations of the extended TIRP. KarmaLego was found faster than H-DFS, IEMiner, and ARMADA (Moskovitch and Shahar, 2015).

However, until KarmaLego (Moskovitch and Shahar 2015, 2015) time intervals mining algorithms were incomplete. In order to discover the complete set of frequent patterns, it is essential to discover the entire set of occurrences of the patterns (within the same entity) (Moskovitch and Shahar, 2015). Previous algorithms referred only to the discovery of the first instance of a symbolic time interval, or TIRP's instance, which means that many instances of potentially frequent TIRPs were not accessed, and as a result extended TIRP's instances were missed and their frequency was lower than it is in practice. Recently Harel and Moskovitch (2021) had introduced TIRPClo, an efficient algorithm for the complete discovery of only the frequent closed TIRPs, a compact subset of all the frequent TIRPs based on which their complete information can be revealed. The algorithm utilizes a memory-efficient index, and a novel method for data projection, which makes it the first algorithm to guarantee a complete discovery of frequent closed TIRPs. This completeness problem exists also with many of the sequential patterns mining algorithms, but, for example, SPADE does it properly. A detailed explanation on the topic of completeness in TIRPs discovery algorithms is in (Moskovitch and Shahar, 2015). Other methods for time intervals mining were proposed, which either do not use Allen's temporal relations (Mörchen, 2006) or use only a subset of these relations (Sacchi et al., 2007).

7 Temporal Patterns Based Classification

Using patterns as features for classification started already with the use of "static" patterns such as itemsets, or association rules, and then experimented also with sequential or time intervals patterns. Frequent pattern mining discovery has been a subject of focus in data mining research with many approaches. For mining various kinds of patterns including itemsets (Liu et al., 1998; Han et al., 2000), sequences (Tseng and Lee, 2009; Fradkin and Mörchen, 2014; Zhou et al., 2016), and TIRPs (Patel et al., 2008; Batal et al., 2012; Moskovitch et al., 2015; Dvir et al., 2020; Schvetz et al., 2021; Novitski et al., 2020; Itzhak et al., 2020). Many association rules based classifiers were proposed by using efficient association rules mining algorithms, such as Apriori (Agrawal and Srikant, 1994), FP-growth (Han et al., 2000), and more. Studies employing frequent sequences for classification were proposed in the last decade (Cheng et al., 2007). Then, typically a feature selection method was applied on the frequent patterns and a model was induced. For that some studies propose suitable features, or in that case sequence, selection methods, such as MMRFS (Cheng et al., 2007) that is based on relevant and redundant measures of the patterns, FeatureMine (Lesh et al., 2000) which is a scalable feature-mining algorithm that uses sequences mining techniques to choose patterns that are useful for classification.

Tseng and Lee (2009) proposed the Classify-By-Sequence (CBS) algorithm to classify large sequence datasets, which classifies by assigning a classification score that consists of combining metrics such as support, confidence, and pattern's length. BIDE-Discriminative (Fradkin and Mörchen, 2014) uses class information and the Information Gain measure for direct mining of predictive sequential patterns in runtime.

Quite simultaneously several studies had proposed the use of time intervals related patterns as features for classification (Patel et al., 2008; Moskovitch and Shahar, 2009). Patel et al. (2008) were the first to use the discovered TIRPs for the classification of multivariate temporal data. They introduced two versions of the IEClassifier, Best_Confidence, in which the class having the highest confidence is selected, while in the Majority_Class, the class to which the majority of the patterns discovered belong is assigned, which outperformed the other classification methods.

Batal et al. (2012) presented the Recent Temporal Pattern (RTP) mining framework, which mines frequent temporal patterns backwards in time, starting from patterns related to the most recent observations. The results shown that the recent TIRPs were more predictive than the other TIRPs. The KarmaLegoSification (KLS) framework (Moskovitch and Shahar, 2015) proposed using TIRPs, after experimenting with several state abstraction methods and number of states. Based on the KarmaLego's complete process of TIRPs discovery, more than binary representation of the TIRPs is enabled. It introduces novel metrics, such as the Horizontal Support which refers to the number of instances discovered for a specific entity (i.e., patient), or Mean Duration, which represents the average duration of the discovered instances in a specific entity (i.e., patient's records). Their results shown that using 3 more general relations, than Allen's relations performed better, and the mean duration was better than the horizontal support. Later the supervised Temporal Discretization for Classification (TD4C) was introduced that shown to outperform SAX and EWD (Moskovitch and Shahar, 2015).

A recent study introduced Maitreya (Moskovitch et al., 2017) a framework that discovers TIRPs only from the cohort of patients having the outcome event. The results shown that representing the TIRPs using the horizontal support outperformed the binary and mean duration representations.

8 Discussion

Heterogeneous multivariate temporal data is becoming increasingly available and accessible in many domains, spanning from internet and cellular usage, through medical data whether from wearable devices, or outpatient and inpatient data, satellites, autonomous cars, and more. This provides significant opportunities for more understanding of various processes in various domains. When using temporal data, several tasks are often of interest: forecasting, in which based on recent series of measurements a future value is predicted; or clustering of time series to find groups of similar time series, for which similarity formulas are required;

classification based on a period of time; or prediction of a future outcome within a given observation time period, based on a time window, using a classifier, continuously. The main focus in most of these tasks is which type of features to extract, for which various types of features were proposed, from explicit features, such as shapelets, or other types of temporal patterns, like Markov chains, sequential patterns, time intervals patterns and more, to deep learning based architectures that learn the "features" by themselves.

Temporal knowledge discovery is another task, in which using various models, and frequent temporal patterns the way processes develop along time can be revealed. A main challenge in such tasks is given a specific event, such as an outcome of interest, to model the data that evolves towards the event, for which visualization is required to facilitate an expert that explores the findings.

A meaningful challenge with multivariate temporal data is their heterogeneity, especially with observational data, which is used for secondary analysis, as shown in Fig. 1. For that a transformation has to be applied, to enable to merge the use of time point values and other types of variables whether represented by events that may or not varying have duration. One of the options is decreasing the granularity of the data along time using methods such as piecewise aggregate approximation, and extracting more than the mean value as commonly done, which will result in multiple series. Another option is to impute data, which is a challenge for itself, especially when there are a lot of missing data, and there is also an option of abstracting the data, as was presented here above using state or gradient abstraction. There is much open room for contributions in the way to analyze multivariate heterogeneous temporal data.

Another important topic in temporal data analytics is temporal knowledge discovery through the discovery of frequent temporal patterns. Most of the methods discover frequent models from univariate temporal data, but other can discover patterns from multivariate temporal data. Sequential patterns mining enables to discover patterns from symbolic events, which is a research field that was intensely researched. A more complicated version that was discussed in this paper is time intervals related patterns discovery, which sequential mining is a private case of. TIRPs discovery was relatively under researched and stores a great potential in representing heterogeneous multivariate temporal data (Fig. 1), after performing temporal abstraction to transform the data into a uniform representation of symbolic time intervals. TIRPs describe the temporal relations among the various variables whether they are represented by raw events that may have varying durations, or symbolic time intervals that are the result of state or gradient abstraction, which may reveal their temporal relations along time. Often there will be interest in TIRPs that end with some event of interest, and there may be several TIRPs like that. This field has still room for contributions. As mentioned earlier, most of these algorithms are incomplete, and the use of TIRPs can be further exploited, beyond the use as features for classification and prediction, which was done mainly so far.

References

Y. Gong, Z. Li, J. Zhang, W. Liu, Y. Zheng, Online spatio-temporal crowd flow distribution prediction for complex metro system, IEEE Transactions on Knowledge and Data Engineering (2020a) 1–1.

Y. Gong, Z. Li, J. yu Zhang, W. Liu, J. Yi, Potential passenger flow prediction: A novel study for urban transportation development, in: AAAI.

Y. Wang, H. Yin, H. Chen, T. Wo, J. Xu, K. Zheng, Origin-destination matrix prediction via graph convolution: a new perspective of passenger demand modeling, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019).

J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, T. Li, Predicting citywide crowd flows using deep spatio-temporal residual networks, Artif. Intell. 259 (2018) 147–166.

Y. Gong, Z. Li, J. Zhang, W. Liu, Y. Zheng, C. Kirsch, Network-wide crowd flow prediction of Sydney trains via customized online non-negative matrix factorization, Proceedings of the 27th ACM International Conference on Information and Knowledge Management (2018).

O. Dvir, P. Wolfson, L. Lovat, R. Moskovitch, Falls prediction in care homes using mobile app data collection, in: AIME.

Y. Shahar, A framework for knowledge-based temporal abstraction, Artif. Intell. 90 (1997) 79–133.

F. Höppner, Time series abstraction methods—a survey, in: GI Jahrestagung.

I. Pratama, A. E. Permanasari, I. Ardiyanto, R. Indrayani, A review of missing values handling methods on time-series data, 2016 International Conference on Information Technology Systems and Innovation (ICITSI) (2016) 1–6.

T. Gueniche, P. Fournier-Viger, V. S. Tseng, Compact prediction tree: A lossless model for accurate sequence prediction, in: ADMA.

R. Moskovitch, C. G. Walsh, F. Wang, G. Hripcsak, N. Tatonetti, Outcomes prediction via time intervals related patterns, 2015 IEEE International Conference on Data Mining (2015) 919–924.

P. Papapetrou, G. Kollios, S. Sclaroff, D. Gunopulos, Mining frequent arrangements of temporal intervals, Knowledge and Information Systems 21 (2009) 133–171.

O. D. Harel, R. Moskovitch, Complete closed time intervals-related patterns mining, in: AAAI.

P. F. Schulam, F. Wigley, S. Saria, Clustering longitudinal clinical marker trajectories from electronic health data: Applications to phenotyping and endotype discovery, in: AAAI.

P. Moran, P. Whittle, Hypothesis testing in time series analysis.

G. Box, G. Jenkins, Time series analysis, forecasting and control.

J. Hamilton, Time series analysis.

S. J. Taylor, B. Letham, Forecasting at scale, PeerJ Prepr. 5 (2017) e3190.

M. Schvetz, L. Fuchs, V. Novack, R. Moskovitch, Outcomes prediction in longitudinal data: Study designs evaluation, use case in ICU acquired sepsis, Journal of biomedical informatics (2021) 103734.

N. Itzhak, A. Nagori, E. Lior, M. Schvetz, R. Lodha, T. Sethi, R. Moskovitch, Acute hypertensive episodes prediction, in: AIME.

P. Novitski, C. M. Cohen, A. Karasik, V. Shalev, G. Hodik, R. Moskovitch, All-cause mortality prediction in T2D patients, in: AIME.

A. Bagnall, J. Lines, A. Bostrom, J. Large, E. J. Keogh, The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Mining and Knowledge Discovery 31 (2016) 606–660.

F. Höppner, Improving time series similarity measures by integrating preprocessing steps, Data Mining and Knowledge Discovery 31 (2016) 851–878.

C. Ratanamahatana, E. J. Keogh, Three myths about dynamic time warping data mining, in: SDM.

T. Rakthanmanon, B. J. L. Campana, A. Mueen, G. E. A. P. A. Batista, M. Westover, Q. Zhu, J. Zakaria, E. J. Keogh, Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping, ACM transactions on knowledge discovery from data 7 3 (2013).

J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures, Data Mining and Knowledge Discovery 29 (2014) 565–592.
J. J. R. Diez, C. Alonso, J. A. Maestro, Support vector machines of interval-based features for time series classification, Knowl. Based Syst. 18 (2005) 171–178.
L. Ye, E. J. Keogh, Time series shapelets: a novel technique that allows accurate, interpretable and fast classification, Data Mining and Knowledge Discovery 22 (2010) 149–182.
A. Kotsifakos, P. Papapetrou, Model-based time series classification, in: IDA.
Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 1735–1780.
A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM 60 (2012) 84–90.
M. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2013) 2796–2802.
H. Deng, G. Runger, E. Tuv, V. Martyanov, A time series forest for classification and feature extraction, Inf. Sci. 239 (2013) 142–153.
A. Bostrom, A. Bagnall, Binary shapelet transform for multiclass time series classification, in: DaWaK.
H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data Mining and Knowledge Discovery 33 (2019) 917–963.
R. Neamtu, R. Ahsan, E. A. Rundensteiner, G. N. Sárközy, E. J. Keogh, H. Dau, C. Nguyen, C. Lovering, Generalized dynamic time warping: Unleashing the warping power hidden in point-wise distances, 2018 IEEE 34th International Conference on Data Engineering (ICDE) (2018) 521–532.
T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: ICLR.
Q. V. Le, T. Mikolov, Distributed representations of sentences and documents, ArXiv abs/1405.4053 (2014).
Y. Goldberg, A primer on neural network models for natural language processing, Journal of Artificial Intelligence Research abs/1510.00726 (2016).
G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Processing Magazine 29 (2012) 82–97.
T. Sainath, A. rahman Mohamed, B. Kingsbury, B. Ramabhadran, Deep convolutional neural networks for LVCSR, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (2013) 8614–8618.
J. Gamboa, Deep learning for time-series analysis, ArXiv abs/1701.01887 (2017).
Z. Yang, M. Dehmer, O. Yli-Harja, F. Emmert-Streib, Combining deep learning with token selection for patient phenotyping from electronic health records, Scientific Reports 10 (2020).
H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science 304 (2004) 78–80.
C. Gallicchio, A. Micheli, Deep echo state network (deepESN): A brief survey, ArXiv abs/1712.04323 (2017).
A. Bagnall, G. Janacek, A run length transformation for discriminating between auto regressive time series, Journal of Classification 31 (2014) 154–178.
H. Chen, F. Tang, P. Tino, X. Yao, Model-based kernel for efficient time series analysis, Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (2013).
D. Fradkin, F. Mörchen, Mining sequential patterns for classification, Knowledge and Information Systems 45 (2014) 731–749.
R. Moskovitch, Y. Shahar, Classification of multivariate time series via temporal abstraction and time intervals mining, Knowledge and Information Systems 45 (2015) 35–74.

I. Batal, D. Fradkin, J. Harrison, F. Mörchen, M. Hauskrecht, Mining recent temporal patterns for event detection in multivariate time series data, KDD : proceedings. International Conference on Knowledge Discovery & Data Mining 2012 (2012) 280–288.
M. Längkvist, L. Karlsson, A. Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, Pattern Recognit. Lett. 42 (2014) 11–24.
Y. Bengio, L. Yao, G. Alain, P. Vincent, Generalized denoising auto-encoders as generative models, in: Proceedings of the 26th International Conference on Neural Information Processing Systems—Volume 1, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013, p. 899–907.
Q. Hu, R. Zhang, Y. Zhou, Transfer learning for short-term wind speed prediction with deep neural networks, Renewable Energy 85 (2016) 83–95.
S. Wang, G. Hua, G. sheng Hao, C. Xie, A cycle deep belief network model for multivariate time series classification, Mathematical Problems in Engineering 2017 (2017) 1–7.
R. Mittelman, Time-series modeling with undecimated fully convolutional neural networks, ArXiv abs/1508.00317 (2015).
D. Banerjee, K. Islam, G. Mei, L. Xiao, G. Zhang, R. Xu, S. Ji, J. Li, A deep transfer learning approach for improved post-traumatic stress disorder diagnosis, 2017 IEEE International Conference on Data Mining (ICDM) (2017) 11–20.
W. Aswolinskiy, R. F. Reinhart, J. Steil, Time series classification in reservoir- and model-space, Neural Processing Letters 48 (2017) 789–809.
F. M. Bianchi, S. Scardapane, S. Løkse, R. Jenssen, Reservoir computing approaches for representation and classification of multivariate time series, IEEE Transactions on Neural Networks and Learning Systems 32 (2021) 2169–2179.
N. Chouikhi, B. Ammar, A. Alimi, Genesis of basic and multi-layer echo state network recurrent autoencoders for efficient data representations, ArXiv abs/1804.08996 (2018).
Q. Ma, L. Shen, W. Chen, J. Wang, J. Wei, Z. Yu, Functional echo state network for time series classification, Inf. Sci. 373 (2016) 1–20.
H. Chen, F. Tang, P. Tino, A. Cohn, X. Yao, Model metric co-learning for time series classification, in: IJCAI.
Z. Che, Decade : A deep metric learning model for multivariate time series.
H. F. Nweke, T. Y. Wah, M. Al-garadi, U. R. Alo, Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges, Expert Syst. Appl. 105 (2018) 233–261.
J. Lin, E. J. Keogh, S. Lonardi, B. Chiu, A symbolic representation of time series, with implications for streaming algorithms, in: DMKD '03.
E. J. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Dimensionality reduction for fast similarity search in large time series databases, Knowledge and Information Systems 3 (2001) 263–286.
R. Moskovitch, Y. Shahar, Classification-driven temporal discretization of multivariate time series, Data Mining and Knowledge Discovery 29 (2015) 871–913.
S. Ramírez-Gallego, S. García, H. Mouriño-Talín, D. Martínez-Rego, V. Bolón-Canedo, A. Alonso-Betanzos, J. M. Benítez, F. Herrera, Data discretization: taxonomy and big data challenge, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 6 (2016).
F. Höppner, Time series abstraction methods—a survey, in: GI Jahrestagung.
K. Ueno, X. Xi, E. J. Keogh, D.-J. Lee, Anytime classification using the nearest neighbor algorithm with applications to stream mining, Sixth International Conference on Data Mining (ICDM'06) (2006) 623–632.
A. Camerra, T. Palpanas, J. Shieh, E. J. Keogh, isax 2.0: Indexing and mining one billion time series, 2010 IEEE International Conference on Data Mining (2010) 58–67.
F. Mörchen, A. Ultsch, Optimizing time series discretization for knowledge discovery, in: KDD '05.
F. Mörchen, Algorithms for time series knowledge mining, in: KDD '06.
S. Kullback, R. A. Leibler, On information and sufficiency, Annals of Mathematical Statistics 22 (1951) 79–86.
M. J. Zaki, Scalable algorithms for association mining, IEEE Trans. Knowl. Data Eng. 12 (2000) 372–390.

J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, D. Yang, H-mine: hyper-structure mining of frequent patterns in large databases, Proceedings 2001 IEEE International Conference on Data Mining (2001) 441–448.
T. Uno, M. Kiyomi, H. Arimura, Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets, in: FIMI.
R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994, p. 487–499.
P. Fournier-Viger, A. Gomariz, M. Campos, R. Thomas, Fast vertical mining of sequential patterns using co-occurrence information, in: PAKDD.
R. Srikant, R. Agrawal, Mining sequential patterns: Generalizations and performance improvements, in: EDBT.
M. J. Zaki, Spade: An efficient algorithm for mining frequent sequences, Machine Learning 42 (2004) 31–60.
J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M. Hsu, Mining sequential patterns by pattern-growth: the prefixspan approach, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 1424–1440.
J. Ayres, J. Flannick, J. Gehrke, T. Yiu, Sequential pattern mining using a bitmap representation, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, Association for Computing Machinery, New York, NY, USA, 2002, p. 429–435.
Z. Yang, M. Kitsuregawa, Lapin-spam: An improved algorithm for mining sequential pattern, 21st International Conference on Data Engineering Workshops (ICDEW'05) (2005) 1222–1222.
P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, R. Thomas, A survey of sequential pattern mining.
W. Gan, C.-W. Lin, P. Fournier-Viger, H. Chao, P. S. Yu, A survey of parallel sequential pattern mining, ACM Transactions on Knowledge Discovery from Data (TKDD) 13 (2019) 1–34.
U. Yun, J. Leggett, WSpan: Weighted sequential pattern mining in large sequence databases, 2006 3rd International IEEE Conference Intelligent Systems (2006) 512–517.
P. Fournier-Viger, R. Nkambou, E. Nguifo, A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems, in: MICAI.
J. Pei, J. Han, W. Wang, Constraint-based sequential pattern mining: the pattern-growth methods, Journal of Intelligent Information Systems 28 (2006) 133–160.
Z. Zheng, Y. Zhao, Z. Zuo, L. Cao, Negative-GSP: An efficient method for mining negative sequential patterns, in: AusDM.
X. Dong, Y. Gong, L. Cao, F-NSP+: A fast negative sequential patterns mining method with self-adaptive data storage, Pattern Recognit. 84 (2018) 13–27.
L. Cao, X. Dong, Z. Zheng, E-NSP: Efficient negative sequential pattern mining, Artif. Intell. 235 (2016) 156–182.
S.-C. Hsueh, M.-Y. Lin, C.-L. Chen, Mining negative sequential patterns for e-commerce recommendations, 2008 IEEE Asia-Pacific Services Computing Conference (2008) 1213–1218.
W. Wang, Negative sequence analysis: A review.
J. Chang, Mining weighted sequential patterns in a sequence database with a time-interval weight, Knowl. Based Syst. 24 (2011) 1–9.
J. Ren, J. Yang, Y. Li, Mining weighted closed sequential patterns in large databases, 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery 5 (2008) 640–644.
C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, A novel approach for mining high-utility sequential patterns in sequence databases, ETRI Journal 32 (2010) 676–686.
G.-C. Lan, T. Hong, V. S. Tseng, S.-L. Wang, Applying the maximum utility measure in high utility sequential pattern mining, Expert Syst. Appl. 41 (2014) 5071–5081.
O. K. Alkan, P. Karagoz, CRoM and HuspEXT: Improving efficiency of high utility sequential pattern extraction, IEEE Trans. on Knowl. and Data Eng. 27 (2015) 2645–2657.

VillafaneRoy, A. HuaKien, TranDuc, MaulikBasab, Knowledge discovery from series of interval events, Journal of Intelligent Information Systems (2000).
J. F. Allen, Maintaining knowledge about temporal intervals, Commun. ACM 26 (1983) 832–843.
F. Mörchen, A better tool than Allen's relations for expressing temporal knowledge in interval data.
R. Moskovitch, Y. Shahar, Fast time intervals mining using the transitivity of temporal relations, Knowledge and Information Systems 42 (2015) 21–48.
P. shan Kam, A. Fu, Discovering temporal patterns for interval-based events, in: DaWaK.
F. Höppner, Learning temporal rules from state sequences.
E. Winarko, J. Roddick, Armada—an algorithm for discovering richer relative temporal association rules from interval-based data, Data Knowl. Eng. 63 (2007) 76–90.
S. yi Wu, Y.-L. Chen, Mining nonambiguous temporal patterns for interval-based events, IEEE Transactions on Knowledge and Data Engineering 19 (2007).
D. Patel, W. Hsu, M. Lee, Mining relationships among interval-based events for classification, in: SIGMOD Conference.
L. Sacchi, C. Larizza, C. Carlo, R. Bellazzi, Data mining with temporal abstractions: learning rules from time series, Data Mining and Knowledge Discovery 15 (2007) 217–247.
B. Liu, W. Hsu, Y. Ma, Integrating classification and association rule mining, in: KDD.
J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, in: SIGMOD '00.
V. S. Tseng, C.-H. Lee, Effective temporal data classification by integrating sequential pattern mining and probabilistic induction, Expert Syst. Appl. 36 (2009) 9524–9532.
C. Zhou, B. Cule, B. Goethals, Pattern based sequence classification, IEEE Transactions on Knowledge and Data Engineering 28 (2016) 1285–1298.
H. Cheng, X. Yan, J. Han, C.-W. Hsu, Discriminative frequent pattern analysis for effective classification, 2007 IEEE 23rd International Conference on Data Engineering (2007) 716–725.
N. Lesh, M. J. Zaki, M. Ogihara, Scalable feature mining for sequential data, IEEE Intell. Syst. 15 (2000) 48–56.
R. Moskovitch, Y. Shahar, Medical temporal-knowledge discovery via temporal abstraction, AMIA ... Annual Symposium proceedings. AMIA Symposium 2009 (2009) 452–6.
R. Moskovitch, F. C. Polubriaginof, A. Weiss, P. Ryan, N. Tatonetti, Procedure prediction from symbolic electronic health records via time intervals analytics, Journal of biomedical informatics 75 (2017) 70–82.

Cloud Big Data Mining and Analytics: Bringing Greenness and Acceleration in the Cloud

Hrishav Bakul Barua and Kartick Chandra Mondal

1 Introduction

Classical data mining algorithms have been very successful in almost all the fields of science and technology ranging from medical analytic or space sciences. The classical concepts of data mining are centered around four main paradigms: Descriptive (Clustering), Associative (Associative Rule Mining), Discriminant (Classification), and Predictive analysis (Regression). However, with the growing amount of data, it is becoming extremely difficult to manage and process it. Retrieving information is also becoming a headache. So, people are moving towards distributed and parallel computing paradigms [8]. But, using such paradigms in a standalone system can be another challenge in terms of scalability and cost-effectiveness, so the concept of data mining in the cloud has come up in the last decade [8]. The paper [8] has summarized the recent applications, trends, techniques, algorithms, and frameworks in cloud data mining and big data mining in the cloud.

Now, is this enough? Certainly not! The way in which data is exponentially increasing, simply harnessing the sea of resources (compute, storage, etc.) available in the cloud even may not be just enough. So, researchers are keen in finding alternatives and effective solutions to this data outburst or we can say "data apocalypse." Graphics Processing Units (GPUs) are the most important hardware assets in this respect. GPUs are a special kind of CPUs which are designed for parallel computing and specifically used to alter memory in a very rapid manner.

H. B. Barua
Embedded Systems and Robotics Research Group, TCS Research & Innovation Labs, Kolkata, India
e-mail: hrishav.barua@tcs.com; hbarua@acm.org

K. C. Mondal (✉)
Department of Information Technology, Jadavpur University, Kolkata, India
e-mail: kartickjgec@gmail.com

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_22

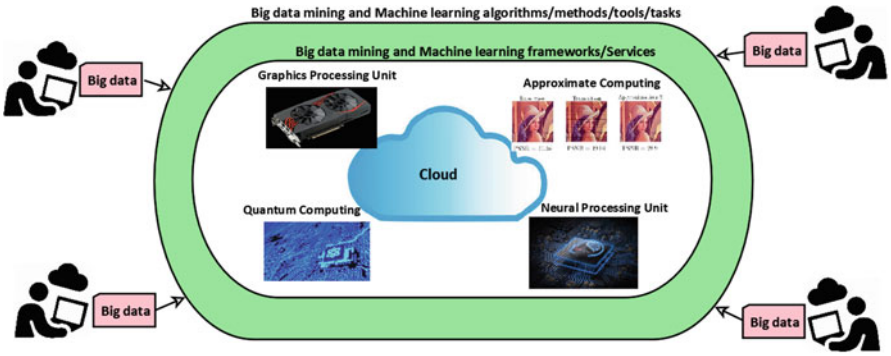


Fig. 1 The general architecture using GPUs, QC, AC, and NPUs for achieving acceleration and greenness in big data mining, analytics, and machine learning

They are designed for processing graphics related calculations in an efficient fashion. GPUs are being used for various data mining and machine learning tasks for accelerating the entire process of analytics and mining [9].

Approximate Computing (AC) or In-exact Computing is a computing paradigm that trades off energy in return of accuracy of results. Some applications which involve visual data processing, audio signal processing, big data mining and such other applications do not need exact computational results most of the time. A bit of deviation in results may always be accepted and awarded in terms of time and energy savings. Some works are well in place which gives the idea of AC and its use in big data analytics and mining related tasks [32, 7].

Quantum Computing (QC) is a buzz word in the today's computing community. The concept of QC is based on the superposition of states of a computer (other than 0 or 1). The quantum computers can solve many computing-intensive problems in a much lesser time than their classical counterparts. Data mining and machine learning techniques are well set to utilize this paradigm as and when it grows and keeps on maturing with time [52]. Its popularity has grown to such an extent that a journal named "Quantum machine intelligence" (on quantum artificial intelligence) [2] has been started by Springer Nature recently.

Figure 1 is a depiction of the cloud-based holistic architecture for big data mining and machine learning using the above discussed technologies. The innermost layer is the cloud computing paradigm in blue color. The middle layer is the combination of cloud-based service and hardware systems consisting of GPUs, NPUs, AC facilities (in the form of hardware and software) and QC facilities (in white eclipse). The outermost layer is the combination of big data and machine learning-related algorithms and tasks for the user to invoke (in green eclipse). Then finally comes the cloud service users which have the big data to be mined or analyzed or use the data for learning purpose.

The main motivation of this chapter lies in the fact that big data is growing even bigger and bigger. Its not convenient to rely entirely on cloud computing resources.

So, we have to rethink the wheel and reinvent the way big data can be managed and processed for the various applications. The current literature do not bring the three above mentioned technologies (GPU, AC, and QC) into one paper for discussion and understanding their capabilities in terms of big data mining and machine learning. We want to put these three mentioned technologies under one umbrella for the researchers to have an easy access and guide of the scenario. This will help the researchers to delve deeper into these areas and come up with efficient data mining for big data and machine learning tasks in future generation cloud systems. We have also commented on the future prospects of these technologies and some other new technologies in the Sect. 6.

This chapter is managed into the following sections. Section 2 gives us a short overview about the concept of data mining in big data paradigm and analytics in cloud environments. The various elements involved in data mining in the big data paradigm has been discussed. We also discuss, why cloud alone is not sufficient to handle the data outburst of today. Section 3 summarizes the recent works related to the use of GPUs in big data mining and machine learning. Section 4 puts forward the concept of AC and its uses in various machine learning and big data analytics related applications. Section 5 gives some recent works on QC and its applications and implementations in big data and machine learning-related algorithms and techniques. Finally, we comment on the three said technologies (AC, QC, and GPUs) and their impact on big data as a whole in Sect. 6. We also talk about future research directions in each of the technologies with some new additions of concepts and methods in cloud related environments. We have also commented on a new technology called Neural acceleration or Neural Processing Units (NPUs). This can be explored to get benefits as per the current literature. The chapter has been concluded with a short summary in Sect. 7.

2 Big Data Mining and Analytics in the Cloud

Data mining has been an important task in many areas of computing for many years. Many multi-disciplinary fields are also highly impacted by data mining and analytics. Data mining simply is the mining of data or discovery of knowledge from structured, semi-structures unstructured data. Now, data analytics is a broader term. It is the process of extraction, cleaning, transforming, modeling, and visualizing the data to find meaningful information and further draw inferences and conclusions out of it. It also involves active machine learning in this process. We define it as follows:

$$Data\ Analytic = Data\ Extraction + Data\ Cleaning + Data\ Transformation + Data\ Modeling + Data\ Visualization + Inference + Conclusion \tag{1}$$

But as the need for more computing power arises, we cannot be sufficient with general computing techniques and methods in our conventional server/workstation

setups. So, the solution to this problem is the paradigm shift towards cloud computing. Data mining in the cloud is the concept of performing data mining tasks in a cloud or cloud-like setup. By cloud-like setup, we mean that we have access to a considerably huge amount of resources in the form of storages, compute nodes, network infrastructure, and other related services. And, this is what we call big data mining in the cloud. The knowledge mined from data is bigger and more complete if the data to be mined is bigger.

$$Big\ Data -> \ Big\ Knowledge + Big\ Intelligence + Big\ Cognition \tag{2}$$

The need for big data mining and analytics is evident in many applications. Mostly, for the applications related to cognition, intelligence, and prediction, data plays a very big role and big data is icing on the cake itself. The benefits of big data can be realized in the form of big knowledge and big Intelligence. But the issues of big data are volume, velocity, variety, and veracity.

But is cloud sufficient to handle these features of big data [8]? We have to use some other technologies other than general cloud resources. We have identified three relevant technologies (GPU, AC, and QC) to achieve efficiency and efficacy in cloud data mining and analytics though we do not claim this is the exhaustive list of the technologies available for this purpose. The different advantages of using these technologies can be represented in Eqs. 3, 4, and 5. Also, some feasible combinations of technologies for future explorations and implementation are shown in Eqs. 6, 7, and 8. Even we can harness the combination of all these three technologies for big data mining, analytics, and machine learning to achieve greenness, acceleration, and efficiency in a cloud-based setup as shown in Eq. 9.

$$Acceleration\ in\ big\ data\ mining -> \ Big\ data\ in\ (GPU + cloud\ computing) \tag{3}$$

$$Greenness\ in\ big\ data\ mining -> \ Big\ data\ in\ (AC + cloud\ computing) \tag{4}$$

$$Efficiency\ in\ big\ data\ mining -> \ Big\ data\ in\ (QC + cloud\ computing) \tag{5}$$

$$(Acceleration + Greenness)\ in\ big\ data\ mining -> \ Big\ data\ in\ (GPU + AC + cloud\ computing) \tag{6}$$

$$(Acceleration + Efficiency)\ in\ big\ data\ mining -> \ Big\ data\ in\ (GPU + QC + cloud\ computing) \tag{7}$$

$$(Efficiency + Greenness)\ in\ big\ data\ mining -> Big\ data\ in\ (QC + AC + cloud\ computing) \quad (8)$$

$$(Acceleration + Efficiency + Greenness)\ in\ big\ data\ mining -> Big\ data\ in\ (GPU + QC + AC + cloud\ computing) \quad (9)$$

3 Graphical Processing Units (GPUs) and Cloud Big Data Analytics

GPUs are specifically designed for graphics related operations. These specialized units have many smaller processing units/threads. They are best suited for doing parallel processing of complex mathematical operations such as matrix operations. Until the recent past, these GPUs have been used for video rendering and multimedia or graphics processing specifically. But, people argued that such powerful processing units may be used for other computations in general rather than just using them in graphical utilities such as video rendering, games, and others. Hence, comes the concept of General-Purpose GPU (GPGPU). Figure 2 shows a typical cloud-based GPU setup to accelerate big data mining and analytics related tasks. We can see, the compute and Data intensive code parts are delegated to the Multi-core GPUs and the other parts are processed in CPUs in the cloud. The GPUs help in accelerating the relevant code segments and process data in a much faster manner.

The paper [54] studies the uses of GPUs in data intensive applications using map-reduce and other graph processing technologies. The authors report their experiences with developing various platforms for data intensive applications and prototypes for the same purpose. Another article [47] is a survey of usage of GPGPUs for cloud frameworks for data intensive computations like big data processing, mining, and analytics. This paper also suggests CPU-GPU based hybrid techniques for future exploration by the interested researchers. The authors think that it has a huge prospect in cloud-based big data processing and analytics activities. Another paper [11] has surveyed the use of GPU based systems and computing paradigms on large-scale general data mining or big data analytics tasks. It discusses the GPU architectures necessary for handling high volume, velocity, and variety of data. Finally, it discusses the limiting causes for proper scalability of such systems in the cloud and some notes on future directions and open research challenges. The forthcoming paragraphs give some of the major contributions on the usage of GPU, Multi-GPU systems and GPGPU systems in big data mining, analytics, and machine learning. Other GPU-CPU hybrid techniques used for big data mining in the cloud or related environments have also been mentioned.

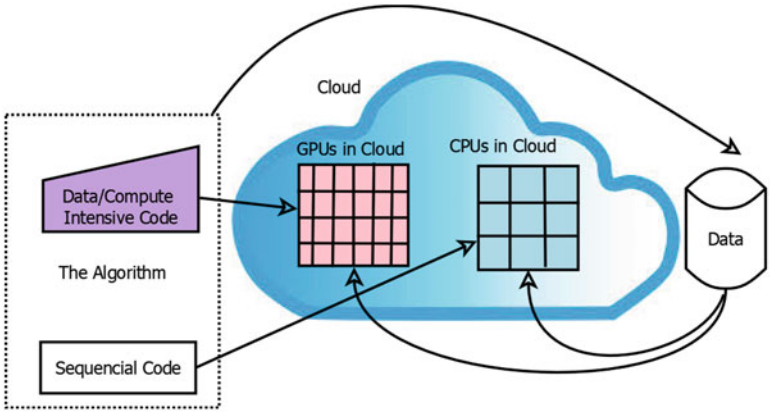


Fig. 2 The general architecture for using GPUs in Cloud Big Data Analytics

In [55], Zhong et al. put forward a GPU based method, G2 for graph processing in the cloud. They have implemented three GPU related optimizations:

- The first one being some APIs to facilitate the gigantic amount of threads in GPU parallelization.
- The second is the introduction of a load balancing method for CPU/GPU architectures using graph partition.
- Thirdly, a memory management system is being incorporated transparently in GPU.

A task scheduling strategy for high throughput in the form of graph tasks using concurrent kernel executions is also implemented. Amazon EC2 virtual cluster of eight nodes is being used to perform experimentation. It shows the efficacy of using GPU acceleration in graph processing tasks in the cloud.

Hai Jiang et al. in [22] puts forward a multi-GPU solution to Map-Reduce (MGMR) which can be used for heavy data processing. This is advantageous over single GPU options too as it eliminates the memory limitations of a single GPU based system. It also avoids atomic operations for acceleration of the entire process. The experimental results have shown the efficacy of these techniques using GPUs in handling huge data in the cloud. Again, [12] gives us an advanced version of MGMR, a Multi-GPU pipelined system (PMGMR) to attend the memory limitations of a simple multi-GPU system and high computational demands for big data. The system has the capacity to use features such as streams and Hyper-Q. PMGMR is 2.5 times more improved and efficient in terms of performance. Due to this, its scalability factor increases a lot and user can use it in a simple manner to write map-reduce related codes effectively. Yet another improvement over the above two GPU based Map-reduce systems is researched in [23]. MGMR++ is optimized for big data processing and analytic. This system even uses hard disks memory when the CPU

and GPU memory gets exhausted. This system also has a 2.5 times improvement in performance compared to MGMR [23].

Authors in [1] propose a very interesting service for the cloud from the perspective of big data and related services. DaaS (Data as a Service) for analytics of real-time data using GPUs is the main contribution of this paper. This system is optimized for network data, customer data, and user data which are obtained from data centers and cloud-based systems. The pre-processing module of the DaaS is GPU enabled for fast and efficient processing and accelerating the entire process. The model is being experimented on huge spatiotemporal data using clustering, self-organizing maps, and neural networks. The promising results suggest that DaaS using GPU can help in achieving SLA (Service Level Agreement) and QoS (Quality of Service) with great efficiency.

The paper [28] is focused on GPUs for machine learning (ML) applications in the cloud. Virtualized GPU techniques have been used as High-Performance Computing (HPC) method. The two main methods referred to are using Hypervisors and vendor-provided virtual GPU technologies. The paper compares results using virtualized GPUs and also by allowing other applications such as 3D-graphics related tasks so as to utilize the computing power in an efficient manner. Some bench-marking ML applications using TensorFlow have been selected for showing scaling between one and multiple GPUs setup. The paper also compares the performance of the two selected virtualization methods. In the end, the paper suggests that it is better (in terms of execution time mitigation) to run ML tasks and other typical GPU tasks together in a mixed fashion rather than running them individually.

A very beautiful comparison of non-GPU and GPU enabled big data clustering process is put forward in [3]. The paper uses a multi-CPU spark system and a multi-GPU TensorFlow enabled system for unsupervised big data learning. The later shows a 5–12 times improvement in time over the former. Jun Wang et al. in [48] states the issues with the big data pre-processing tasks. These tasks typically take huge time and computing power in identifying the multi-level hierarchy in big data. Also, current data have very high dimensionality and it is difficult to scale learning algorithms in such a situation. So, there is an ardent requirement to device a scalable solution to create a tree structure for big data. Incremental K-means is used to serve this purpose. Dimensionality reduction is also incorporated as a part of pre-processing. The underlying architecture used is CUDA (Compute Unified Device Architecture), so as to facilitate big data requirements. It has proved efficient with real-world data after visualization using a dendrogram.

The research in [25] captures the problem of scalability in Evolutionary Decision Trees (DT). It is a hybrid approach of CPU+GPU computing, where the tree structure is searched in a sequential manner in CPU and the fitness is calculated in GPU. As a result, the process is accelerated by thousand times using 4 GPUs while using a data-set of about 1 billion objects. Youcef Djenouri et al. [15] exploit the prospects of GPU cluster computing. Frequent item-set mining in a single scan has been chosen for the experiment to reduce time complexity. The authors have proposed a total of three HPC based techniques for the purpose. The first one uses

a GPU based method to efficiently map thread blocks to input blocks. In the second method, a cluster architecture is used to schedule jobs independently to workers belonging to that cluster. The third one is a multi-cluster architecture consisting of GPUs to accelerate the frequent item-set mining task. Apart from these, several strategies have been incorporated for GPU thread to reduce divergence and load imbalance in clusters. The third technique has been seen to outperform the first and the second one in speedup parameters, specifically 350 times faster for low minimum support from big data perspective.

The paper [5] puts up an interesting method of road traffic incident prediction using a huge amount of data with deep learning methods on GPU platforms. The deep learning method uses three different types of data: road traffic-related data, vehicle detector station, and incident data. These data are taken from California Department of Transportation (Caltrans) Performance Measurement System (PeMS). The research published in [13] focuses on the utility of big data mining on the Internet of Things (IoT) related applications. The authors have used a GPU-aware architecture using Histogram-based segmentation of moving objects. The method takes into account the pixel oriented approach pertaining to GPUs (called pixHMOS_gpu). The experimentation has proved the efficacy of this technique bridging the gaps of computational bottleneck often associated with IoT based systems for big data processing and mining.

Patryk Orzechowski et al. [34] give an efficient bi-clustering method for high volume big data using GPUs. Evolutionary search-based bi-clustering (EBIC) has been implemented successfully using multi-GPUs for achieving high scalability. The applicability of this method can be found in RNA-sequencing related experiments. The paper [18] uses CUDA enabled GPUs for performing complex binary bi-clustering in data mining applications. The authors present CUBiBit to accelerate the binary bi-clustering tasks by using CPUs and GPUs and CUDA architectures. The experimental results depict its amazing speedup of 116 compared to the current method BiBit having 16 CPU cores and three NVIDIA K20 GPUs. GPU based CUDA processing architectures for spatial data mining is discussed in [33]. The spatial data here consists of spatial and non-spatial attributes. So, it is difficult to process and mine such data with general processors or even parallel processing units. The author has proposed a CUDA based architecture for the same where the experiments have been conducted on TIGER/Line data from US census. The results have displayed the superiority of this technique for spatial data mining applications.

4 Approximate Computing (AC) and Cloud Big Data Analytics

Approximate computing or In-exact computing is a technique for trading off result accuracy with speed and energy. It has been successfully used in many domains ranging from machine learning to financial data analysis. Big data is a very strong

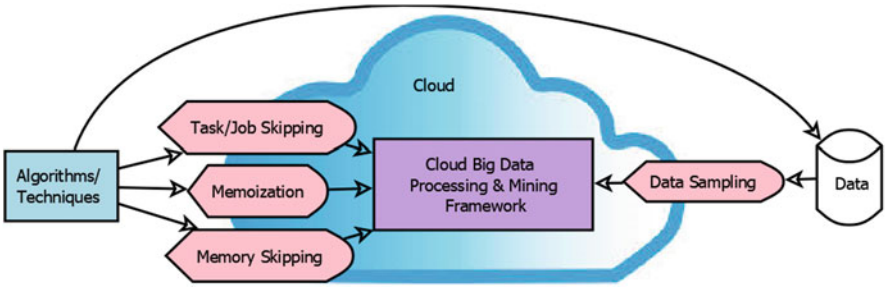


Fig. 3 The general architecture for using AC in Cloud Big Data Analytics

and important target for AC as big data does not always require exact analytics output but needs a summary of the output. Losing a few data items of big data while performing analytics will not impact the final result of mining and analytics. A very small change in the data often does not have the caliber to change the meaning of the predicted value or shift its importance [32]. Some recent articles [32, 7] state the techniques and applications of AC used in big data and machine learning perspective in the cloud and related distributed platforms. The remainder of this section discusses the usage and applications of AC in big data mining and machine learning in cloud environments. Figure 3 gives an idea of the architecture for using AC techniques atop cloud data mining or analytics frameworks. The major techniques are task/job skipping, memoization, and memory skipping. Data sampling can also be applied to cut down unnecessary data and use only a small subset of it for further processing without much change in the mining output.

Shuai Ma et al. [31] put forward the case of big data analytics and its applications in today's world. Big data analytics need huge computational and storage power. To harness the optimal solutions from big data it is necessary that a huge amount of computing is devoted. But in that case, the efficiency of the systems may degrade as the data increases. So, if the optimal solution is replaced with some "good enough" solution which is acceptable than it is a desirable situation for all. Techniques such as approximate query processing and data approximations are being employed in big data analytics. Basically, the query approximation technique has been used in pattern matching in graphs, compression of trajectory and computations of dense sub-graph. Moreover, the data compression technique has been used successfully in shortest path computation, network anomaly detection, and link prediction in graphs for social media analytics.

Barua et al. [6] have proposed a method using loop perforation technique of AC in association rule mining. The new approach cuts down the execution time of the mining task with a loss of accuracy in finding the rules which is acceptable in case of big data volumes. This method can be implemented in parallel systems such as a cloud to attain more efficiency and performance improvement. In [4], authors address the issue of dependency on big data mining results on data variety and its impact on using AC. Using AC when the data is big but variety is low is a

trivial task as per the techniques available nowadays such as sampling, memoization, task skipping, etc. But the same thing, if used in high variety big data-sets, can be disastrous as skipping data from a certain part of such big data may skip a whole lot of a variety from it causing unacceptable results. So, the paper [4] has proposed Gapprox, which is a data variety aware system for approximate computing. The method used is a sampling technique after clustering the data into relevant buckets using intra/inter-cluster distance. The size of the blocks and samples are optimized for Quality of Result and acceptable confidence and error bound. The experimental results show that it outperforms ApproxHadoop [17] by 17x speedup and Sappox [53] by 8x with 5% error tolerance by a user with 95% confidence.

The authors of [17] have put forward an approximate version of the map-reduce paradigm (ApproxHadoop) using task skipping or dropping and data sampling methods. The error bounds for various map-reduce programs have been theoretically determined using statistical formulas. The major applications targeted are data analytics, scientific computation, and machine learning. The speedup achieved is about 32x if the user is ready to tolerate an error of 1% having 95% confidence.

Xuhong Zhang et al. [53] have put forward an approximation method for arbitrarily chosen sub-data-sets from large data-sets. Generally, if the sub-data-sets are uniformly distributed, sampling works pretty well. But for unevenly distributed sub-data-sets the sampling efficiency is very low resulting in poor accuracy in the estimation. For this reason, a distribution-aware method is required. The logical partition of a data-set having sub-data-sets is examined for occurrences. This information is used for online sampling. Hadoop is used to implement this method called Sapprox. The results show that it is 20x times faster than its precise version.

A novel approach to approximate computing using a neural network (NN) architecture is presented in [35]. The said NN architecture is a combination of two NNs, one being the approximator and another is the predictor. The approximator gives us the approximate version of the results. The predictor does the quality check by determining whether the input data is eligible for approximation for a given output accuracy requirement for any application. It is not easy to create such a combined NN having two different NNs as there will be coordination issues and different optimization objectives. So, AXNet is a combination of approximator NN and predictor NN to create a new holistic NN having end-to-end trainable capabilities. Multi-task learning is being used in AXNet to find a better and higher number of approximable candidate samples. The error due to approximation is minimized and training cost is mitigated too. The experimentation with this novel NN architecture shows its efficacy in reducing training time and determining approximable samples.

The usage of Spiking Neural Networks (SNNs) for data intensive applications such as analytics and vision is discussed in [44]. It is difficult to find adequate compute and storage power for large-scale SNNs. Hence, the paper proposes AxSNN atop the already proposed parallel version of the same. It is being used to improve the computational efficiency of SNNs in general. SNNs work with inputs and outputs of neurons which are generally represented as time series of spikes. The internal states of a neuron are updated by spikes at the output of a neuron

which is connected to it. AxSNN leverages the notion of approximate computing by skipping neuron updates triggered by spikes considering that it has minimum impact on the quality of output. This is done to improve the computing energy and memory efficiency. Parameters that are considered for approximating neurons are average spiking rates, current internal states of neurons and the weights of synaptic connections. Approximate computing on a neuron is attained by making it sensitive to a subset of its inputs and sending spikes to a subset of its outputs. The overall system is monitored and approximation modes are updated such as the energy savings are optimized and quality loss is minimum. It has been tested in the form of hardware and software implementations both. These have been tested in many image recognition applications and it achieves 1.4–5.5x reduction in operations. It is equivalent to a 1.2–3.7x reduction in energy approximately in hardware and software implementations without any loss of output.

The paper [19] puts forward a method for approximating Convolution Neural Networks (CNNs) for image processing, big data analytics, computer vision, mining, AI and ML related applications. The paper proposes a systematic method for checking the error tolerance characteristics of a deep CNN and determines the parameters from the set of parameters which can be targeted to improve speed of the network in the inference stage. The idea is to cut down the filters as per their significance in a convolution layer. This enables a trade-off between output quality and speedup.

Authors in [43] give us a methodology for introducing efficient approximate computing in Recurrent Neural Networks (RNNs)—Long Short Term Memory (LSTM). LSTMs are generally used for text generation, speech recognition, and related application areas. Generally, such applications in a big data perspective can be computation-intensive in nature. It is difficult to address such issues with distributed, parallel cloud computing setups. To overcome these constraints, the authors have proposed AxLSTM (Approximate LSTM). AxLSTM is being used in sequence-to-sequence learning using TensorFlow deep learning framework. It achieves a speedup of about 1.31x maximum with no loss of accuracy of output and 1.37x speedup when acceptable reductions in output quality are permitted.

Do Le Quoc et al. [36] put forward a data analytics system for stream processing with privacy preservation. PRIVAPPROX is a system which gives us a low latency stream analytics with proper privacy preservation for users. There are many features of this framework, but the most important one being the ability to determine the optimum trade-offs between output quality with minimizing error and the execution cost. It is also reported to have the ability to do real-time stream analytic using distributed system setups. The main idea of this framework is the combination of sampling and randomized response. Due to this combination of two unique features, it is a system with high privacy for users and high performance in terms of execution time and energy savings.

The article [37] aims at presenting a stream processing system with less computation and response time achieved by AC. The authors have also taken care of error bounds so that the system does not fail with excessive approximation. The algorithm is designed in such a way that it can be used with Apache Spark

batched Streaming and pipeline based Apache Flink as well. StreamApprox is the full prototype of the described system atop Apache Spark Streaming and Apache Flink. The experimentation on real-world case studies shows that it has a speedup of at least 1.2x and a maximum of 3x compared to the classical counterparts.

The paper [50] targets approximate computing for stream analytics in real-time for IoT devices. The paper proposes APPROXIOT which employs an online hierarchical classified reservoir sampling technique that produces approximate output with defined error bounds. Edge computing resources are being used to realize this technique. Apache Kafka is being used as the underlying framework to implement this method. A set of real-world case studies have been used for evaluation, which shows that it attains a speedup of 1.3x–9.9x when the sampling fraction is being changed from 80% to 10%.

The work in [27] gives a data analytics method for incremental approximate computing. Incremental computing is based on memoization of intermediate results in a multi-step process where the intermediate output is memorized for sub-computations. AC means skipping jobs or tasks or sampling huge data-sets into sub-data-sets for efficient use. The paper explores these two methods by designing a sampling algorithm that selects samples on the basis of memoized data from previous runs. It is based on a self-adjusting computation that produces output with error bounds. It is termed as IncApprox and is based on the Apache Spark Streaming framework. The experimentation on real-world case studies confirms the system to have used incremental and approximate computing delivering high-efficiency benefits.

5 Quantum Computing (QC) and Cloud Big Data Analytics

The concept of Quantum computing [39, 14, 30] emerged in 1980. The first quantum mechanical model was described by Paul Benioff, showing that theoretically there is the full possibility of such a computer. The Quantum computing paradigm is not restricted to the two conventional states of a classical computer, i.e., "0" and "1." A quantum computing setup can have a superposition of such states which are called qubits. Quantum computing has the ability to solve some computing-intensive problems in a much faster manner than their classical counterparts. The paper [45] reviews the utility of quantum computing in big data processing and machine learning-related applications and its current scenario in the research community. The paper discusses Quantum Artificial Neural Networks which we may call as QANN. It can speed up the learning process of any conventional neural network by many folds. The use of quantum computing in supervised and unsupervised learning is being discussed. Big data analytics can be highly improved in terms of speed-up with the help of quantum computing. The paper also discusses the challenges and future prospects of quantum computing in machine learning and related fields.

The paper [42] surveys the idea of implementing classical machine learning algorithms or their specific computation hungry sub-parts in quantum setups. Most

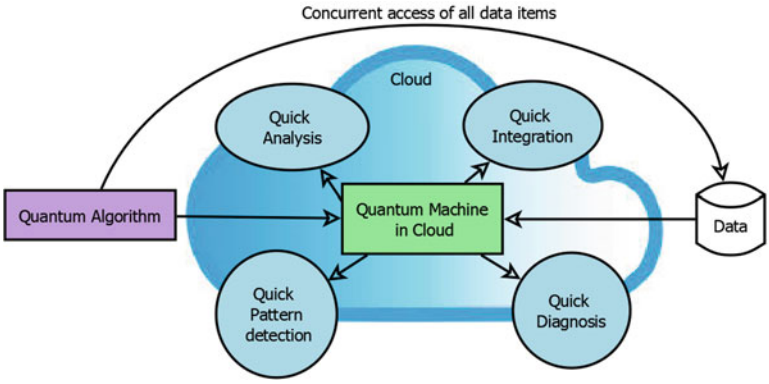


Fig. 4 The general architecture of using QC in Cloud Big Data Analytics

of the stochastic methods can be designed in a way that it can be implemented in quantum systems. The paper gives a very clear picture of quantum machine learning and puts forward the high-level descriptions of the existing methods and techniques along with the low-level technicalities. The paper also comments on the future and research prospects of quantum learning theory. Figure 4 conceptualizes the idea of QC accelerated mining and analytics in the cloud. The basic idea is the use of quantum algorithms on quantum machines in cloud setup which can concurrently access the data in hand for processing, analysis, integration, and pattern detection.

The paper [10] is a review article that speaks about the promises of quantum techniques to solve machine learning problems and data pattern mining applications more efficiently than classical systems. This is due to the fact that the quantum computing paradigm is ought to work in such a way that it can outperform classical systems in certain tasks specifically. The new field that has been talked about in this article speaks about Quantum Machine Learning (QML) that is dedicated towards implementing quantum algorithms and techniques and software which can model machine learning systems and programs. But the real challenge in such a scenario is the hardware that can support quantum systems and software.

An idea about how quantum computing can be utilized for big data-related technologies is shown in [49]. Big data has seen a tremendous explosion in recent years and it is becoming bigger and bigger with each passing day. With such a situation it is not possible to rely simply on cloud resources such as parallelism, distribution, huge compute power, and storage alone for the sake of better and efficient analytics. The main concept of quantum computing allows it to perform quantum parallelism and can be very fast as compared to classical systems even with cloud support. The paper discusses the Grover search algorithm. Quantum machine learning can be used to implement big data analytics through the eye of quantum computation. The paper discusses the main applications of quantum computation in data mining and related tasks.

Patrick Rebentrost et al. [38] discuss the implementation of a supervised learning technique in quantum computing architecture. Support Vector Machine (SVM) has been chosen for this experiment and good results have been achieved. Generally, the SVM takes polynomial time complexity in a classical computer. It has been implemented in logarithmic time complexity in quantum setup displaying an exponential speedup in the process. This big data related SVM technique is based on the non-sparse matrix exponentiation technique. Another research work [20] discusses the limitations of a machine learning problem considering SVM specifically. The feature space determination and its size are the real problems in machine learning and these need adequate attention. The quantum algorithms have quantum state spaces using entanglement and interference which are really large if compared to classical counterparts. The paper proposes algorithms on the basis of quantum state spaces which can be used as feature space for ML problems. The quantum variational classifier has a variational quantum circuit and the quantum kernel estimator can optimize a classical SVM by estimating the kernel function on the quantum setup.

In [41] the prospects of quantum computing in pattern classification of big data have been discussed. The conventional machine learning algorithms can be enhanced in terms of performance using quantum information theories. The paper introduces the quantum pattern classification algorithm and its application in handwritten digit recognition from the MNIST data-sets. Article [40] gives us a quantum version of K-nearest neighbors (QKNN) using a metric of Hamming distance. A quantum circuit is being created and implemented to calculate Hamming distances in testing samples and the feature vectors in the training set. The QKNN method achieves good performance in terms of complexity and also shows good classification accuracy. This method has proven itself to be better than many existing methods.

The paper [46] discusses the idea of offloading a quantum machine learning task from a classical computer to a remote quantum computer with proper data privacy preservation methods implemented atop it. Distributed secure quantum machine learning (DSQML) is a method which is proposed to do the said job. It not only offloads ML tasks but also accesses remote data in the database server. A robust protocol is designed to prevent any eavesdropping or any hindrance in the learning process. The protocol is made for the classification of high dimensional vectors and can be a good candidate to be used in big data applications in the future.

Ashish Kapoor et al. [26] state that the use of quantum computation methods can bring about a drastic improvement in the computational complexity of perceptron learning. The paper gives two algorithms, one uses quantum information processing and another shows that the mistake bound of classical computing can be improved by quantum computing. Zhaokai Li et al. [29] show a quantum parallelism based machine learning algorithm to implement handwriting recognition. The quantum machine which is used for this experiment has a 4-qubit NMR test bench. In today's age of artificial intelligence and big data, such a quantum-based method could be very lucrative in the long run.

The paper [21] gives us a quantum computing approach for image processing and computer vision-based applications like object detection. Object detection is one of the most important applications in many areas such as surveillance, robotics, and many more. The research in this paper presents an automated object detection method using quantum techniques. The experimentation results have proved the supremacy of such a quantum-based algorithm in object detection accuracy which minimizes measurement errors in general.

Matthew C. Johnson et al. [24] have proposed the marriage of quantum computing devices and distributed computing paradigms, which can be thought of as a service in the cloud as well. The system has APIs and data models for quantum computing. Some software are also in place for creating these quantum data models and compute them to get the relevant results from the quantum devices. This way the distributed computing paradigms can be made more effective for high performance by integrating quantum device into their setups.

6 Discussions, Prospects, and Future Trends

The chapter has categorized literature into three primary buckets for the three primary game-changing technologies. However, we want to have a better insight into the technologies and trends from the lens of big data, cloud data mining, and machine learning. We also commented on the future of the three technologies along with NPUs which have not been discussed extensively in the chapter like the other three.

Table 1 gives us a first-hand list of all the literatures cited in this chapter on the basis of the discussed technologies for acceleration (GPU and QC) and greenness (AC) and application areas in cloud and related environments. It shows that GPU and AC have dominated the big data mining and analytic area in recent times as compared to machine learning. But, QC has more impact on machine learning than big data mining. So, a combination of these technologies can be thought of as new

Table 1 A classification based on different technologies used for big data mining and machine learning in cloud

Technology used	Application area	References
GPU	Big data mining	[54, 47, 11, 9, 55, 22, 12, 23, 1, 3, 48, 25, 15, 13, 34, 33, 18]
	Machine learning	[9, 28, 3, 5]
AC	Big data mining	[32, 7, 37, 6, 17, 53, 4, 27, 36, 50, 31]
	Machine learning	[44, 19, 43, 35]
QC	Big data mining	[52, 45, 49, 40, 24]
	Machine learning	[52, 42, 10, 49, 38, 41, 46, 20, 26, 29, 40, 21, 24]

avenues for researchers to explore and architect frameworks and platforms for big data and machine learning.

Approximate Computing: AC has been instrumental in reducing time complexity for many cloud and non-cloud operations specifically for big data and machine learning applications. The different neural network architectures are the prime targets of AC in machine learning as seen in the literature. The targeted algorithms and tasks are stream processing, association rule mining, map-reduce paradigm, Hadoop ecosystem, spark framework, streaming data analytic, spiking neural networks, convolution neural networks, Kafka framework, real-time analytic, recurrent neural networks, and long short term memory. The common thing seen in this technology is that it can considerably accelerate the analytics, mining, or learning process with energy savings to a great extent. The error bound is also taken care of by various statistical models and allowable to a safe extent. In the future, researches can be focused on implementing approximate computing services in cloud setups, AxCaaS- Approximate Computing as a Service [7]. One more area to give focus can be from big data and machine learning perspective, by creating approximate computing facilitated big data analytic platforms or services in cloud.

Accelerated GPU: GPUs have been a powerful hardware tool for acceleration in any cloud-based cluster systems or even in a standalone system. Generally they are used for multimedia computing and video/image processing applications. But, researchers argued that they can be of great help in general-purpose computing too. Hence, we have a different class of GPUs called GPGPUs or General-Purpose GPUs. The different services and algorithms targeted by GPUs are graph processing and mining, large-scale data mining, map-reduce paradigm, GPU based Data as a service (DaaS) in the cloud, real-time data analytics, neural networks, self-organizing maps, clustering, machine learning in 3D graphics, spark framework, TensorFlow system, unsupervised learning, k-means, decision trees, deep learning, data fusion, bi-clustering, and bio-informatics. The surveyed literature in this chapter tells that graph processing and mining is the most suitable candidate for implementing GPU based acceleration. The main three concerns of big data analytics and related technologies are big volume, big velocity, and big variety and GPUs are capable of taking care of all these three elements of big data efficiently. The main advantage of GPUs can be stated as the achievement of service level agreements and quality of service in cloud-based systems. Here big data is being harnessed as workflows by various users. So, we want to suggest GPU as a service specifically for big data scenarios (BD-GPUaaS) and machine learning in GPU as a service (ML-GPUaaS) in the cloud for advanced machine learning techniques such as deep learning.

Quantum Computing: QC is in the focus light for the last few years in many domains of computing and the major reason is its speed of computing some of the conventional things in an un-imaginable manner compared to classical counterparts. Quantum computing has been proposed by many researchers for big data and machine learning-related computations. Quantum computing has

been widely used to implement machine learning techniques in general. The theory of quantum learning and information processing has been well crafted to suit the requirements of these applications. The various targeted areas are artificial neural networks, support vector machine, pattern classification, k-nearest neighbors, object detection and recognition. Some literatures also talk about quantum computing in a distributed environment such as cloud. There they can offload quantum computing-related tasks to quantum computers and other tasks can be done in the in-house system itself. We want to suggest quantum computing as a service for big data and machine learning for the future. QCaaS has been mentioned in the paper [8] and we hope that it can be a good platform for the researches to conduct fruitful researches on big data and machine learning. We can also think about BD-QCaaS and ML-QCaaS for big data and machine learning in the future to achieve acceleration and performance improvement in the future.

Neural Processing Unit: We would also like to give a possible future thought on a new technology that can be used in this respect specifically in machine learning. The rise of neural and AI accelerators or Neural processing units (NPUs) are in the process. The neural accelerators [32, 16, 51] are best suited for neural network computations for machine vision, big data processing and learning from a huge amount of data. The application areas can be robotics, IoT, and data intensive computations at the edge/fog devices. So, we strongly recommend this technology to be actively researched and implemented in cloud-based systems for big data mining and machine learning. The neural accelerators for AI applications are inherently built to approximate the regions of code by changing them to neural models. So, indirectly these accelerators are using the AC paradigm and researchers who are working with AC currently can also look into this new area of research and development from big data and machine learning perspective.

7 Conclusions

This chapter provides a very concise and lucid depiction of some of the recent technologies which can enhance the cloud or related high-performance computing models. This will support the efficiency issue of big data mining and analytics or machine learning-related tasks. The chapter gives a generic overview of data mining from a big data perspective. We have defined big data as an entity consisting of 4Vs (volume, velocity, variety, and veracity) but on a larger scale. The cloud computing paradigm is being used successfully for the last decade to facilitate big data analytic and mining, to bring efficiency and performance in the entire analytic process. Existing surveys and researches have conveyed that having cloud resources at our disposal does not help in achieving the maximum speedup and complexity benefits in terms of big data. We have identified three future technologies to solve this situation in a cloud platform. Two major kinds of performance aware

technologies are identified such as Greenness and Acceleration in computation. Under the umbrella of green computing, we chose approximate computing (AC) and within acceleration, we have chosen GPU based acceleration and Quantum computing (QC) based performance acceleration. We have also shown that Neural Processing Units (NPUs) can also prove beneficial for big data and machine learning process acceleration for future generation cloud systems.

References

1. John Olorunfemi Abe and Burak Berk Ustundaug. A data as a service (DaaS) model for gpu-based data analytics. *arXiv preprint arXiv:1802.01639*, 2018.
2. Giovanni Acampora. Quantum machine intelligence. https://www.springer.com/journal/42484, 2019, 2020. [Online; accessed 25-Sep-2020].
3. Widiarto Adiyoso, Adila Krisnadhi, Ari Wibisono, Sumarsih Condroayu Purbarani, Anindhita Dwi Saraswati, Annissa Fildzah Rafi Putri, Ibad Rahadian Saladdin, and S Reyneta Carissa Anwar. Time performance analysis of multi-CPU and multi-GPU in big data clustering computation. In *2018 International Workshop on Big Data and Information Security (IWBIS)*, pages 113–116. IEEE, 2018.
4. Hossein Ahmadvand, Maziar Goudarzi, and Fouzhan Foroutan. Gapprox: using Gallup approach for approximation in big data processing. *Journal of Big Data*, 6(1):20, 2019.
5. Muhammad Aqib, Rashid Mehmood, Ahmed Alzahrani, and Iyad Katib. In-memory deep learning computations on GPUs for prediction of road traffic incidents using big data fusion. In *Smart Infrastructure and Applications*, pages 79–114. Springer, 2020.
6. Hrishav Bakul Barua and Kartick Chandra Mondal. Green data mining using approximate computing: An experimental analysis with rule mining. In *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 115–120. IEEE, 2018.
7. Hrishav Bakul Barua and Kartick Chandra Mondal. Approximate computing: A survey of recent trends—bringing greenness to computing and communication. *Journal of The Institution of Engineers (India): Series B*, pages 1–8, 2019.
8. Hrishav Bakul Barua and Kartick Chandra Mondal. A comprehensive survey on cloud data mining (CDM) frameworks and algorithms. *ACM Computing Surveys (CSUR)*, 52(5):1–62, 2019.
9. K Bhargavi and B Sathish Babu. Accelerating the big data analytics by gpu-based machine learning: A survey. In *International Symposium on Sensor Networks, Systems and Security*, pages 63–83. Springer, 2017.
10. Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.
11. Alberto Cano. A survey on graphic processing unit computing for large-scale data mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(1):e1232, 2018.
12. Yi Chen, Zhi Qiao, Spencer Davis, Hai Jiang, and Kuan-Ching Li. Pipelined multi-GPU MapReduce for big-data processing. In *Computer and Information Science*, pages 231–246. Springer, 2013.
13. Alfredo Cuzzocrea and Enzo Mumolo. A novel gpu-aware histogram-based algorithm for supporting moving object segmentation in big-data-based IoT application scenarios. *Information Sciences*, 496:592–612, 2019.
14. Vasil S Denchev and Gopal Pandurangan. Distributed quantum computing: A new frontier in distributed systems or science fiction? *ACM SIGACT News*, 39(3):77–95, 2008.

15. Youcef Djenouri, Djamel Djenouri, Asma Belhadi, and Alberto Cano. Exploiting gpu and cluster parallelism in single scan frequent itemset mining. *Information Sciences*, 496:363–377, 2019.
16. Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Neural acceleration for general-purpose approximate programs. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 449–460. IEEE, 2012.
17. Inigo Goiri, Ricardo Bianchini, Santosh Nagarakatte, and Thu D Nguyen. ApproxHadoop: Bringing approximations to MapReduce frameworks. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 383–397, 2015.
18. Jorge González-Domínguez and Roberto R Expósito. Accelerating binary biclustering on platforms with CUDA-enabled GPUs. *Information Sciences*, 496:317–325, 2019.
19. Muhammad Abdullah Hanif, Rehan Hafiz, and Muhammad Shafique. Error resilience analysis for systematically employing approximate computing in convolutional neural networks. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 913–916. IEEE, 2018.
20. Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.
21. Ling Hu and Qiang Ni. Quantum automated object detection algorithm. In *2019 25th International Conference on Automation and Computing (ICAC)*, pages 1–4. IEEE, 2019.
22. Hai Jiang, Yi Chen, Zhi Qiao, Kuan-Ching Li, Wonwoo Ro, and Jean-Luc Gaudiot. Accelerating MapReduce framework on multi-GPU systems. *Cluster Computing*, 17(2):293–301, 2014.
23. Hai Jiang, Yi Chen, Zhi Qiao, Tien-Hsiung Weng, and Kuan-Ching Li. Scaling up MapReduce-based big data processing on multi-GPU systems. *Cluster Computing*, 18(1):369–383, 2015.
24. Matthew C Johnson, David AB Hyde, Peter McMahon, Kin-Joe Sham, and Kunle Tayo Oguntebi. Integration of quantum processing devices with distributed computers, November 19 2019. US Patent 10,484,479.
25. Krzysztof Jurczuk, Marcin Czajkowski, and Marek Kretowski. Multi-GPU approach for big data mining: global induction of decision trees. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 175–176, 2019.
26. Ashish Kapoor, Nathan Wiebe, and Krysta Svore. Quantum perceptron models. In *Advances in Neural Information Processing Systems*, pages 3999–4007, 2016.
27. Dhanya R Krishnan, Do Le Quoc, Pramod Bhatotia, Christof Fetzer, and Rodrigo Rodrigues. Incapprox: A data analytics system for incremental approximate computing. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1133–1144, 2016.
28. Uday Kurkure, Hari Sivaraman, and Lan Vu. Machine learning using virtualized GPUs in cloud environments. In *International Conference on High Performance Computing*, pages 591–604. Springer, 2017.
29. Zhaokai Li, Xiaomei Liu, Nanyang Xu, and Jiangfeng Du. Experimental realization of a quantum support vector machine. *Physical review letters*, 114(14):140504, 2015.
30. Hoi-Kwong Lo, Tim Spiller, and Sandu Popescu. *Introduction to quantum computation and information*. World Scientific, 1998.
31. Shuai Ma and Jinpeng Huai. Approximate computation for big data analytics. *arXiv preprint arXiv:1901.00232*, 2019.
32. Ravi Nair. Big data needs approximate computing: technical perspective. *Communications of the ACM*, 58(1):104–104, 2014.
33. Byoung-Woo Oh. Parallel algorithm for spatial data mining using CUDA. *JOURNAL OF ADVANCED INFORMATION TECHNOLOGY AND CONVERGENCE*, 9(2):89–97, 2019.
34. Patryk Orzechowski and Jason H Moore. Ebic: a scalable biclustering method for large scale data analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 31–32, 2019.

35. Zhenghao Peng, Xuyang Chen, Chengwen Xu, Naifeng Jing, Xiaoyao Liang, Cewu Lu, and Li Jiang. AXNet: Approximate computing using an end-to-end trainable neural network. In *Proceedings of the International Conference on Computer-Aided Design*, pages 1–8, 2018.
36. Do Le Quoc, Martin Beck, Pramod Bhatotia, Ruichuan Chen, Christof Fetzer, and Thorsten Strufe. Privacy preserving stream analytics: The marriage of randomized response and approximate computing. *arXiv preprint arXiv:1701.05403*, 2017.
37. Do Le Quoc, Ruichuan Chen, Pramod Bhatotia, Christof Fetzer, Volker Hilt, and Thorsten Strufe. StreamApprox: approximate computing for stream analytics. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, pages 185–197, 2017.
38. Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.
39. Eleanor Rieffel and Wolfgang Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys (CSUR)*, 32(3):300–335, 2000.
40. Yue Ruan, Xiling Xue, Heng Liu, Jianing Tan, and Xi Li. Quantum algorithm for k-nearest neighbors classification based on the metric of hamming distance. *International Journal of Theoretical Physics*, 56(11):3496–3507, 2017.
41. Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. Quantum computing for pattern classification. In *Pacific Rim International Conference on Artificial Intelligence*, pages 208–220. Springer, 2014.
42. Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.
43. Sanchari Sen and Anand Raghunathan. Approximate computing for long short term memory (LSTM) neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2266–2276, 2018.
44. Sanchari Sen, Swagath Venkataramani, and Anand Raghunathan. Approximate computing for spiking neural networks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 193–198. IEEE, 2017.
45. Tawseef Ayoub Shaikh and Rashid Ali. Quantum computing in big data analytics: A survey. In *2016 IEEE International Conference on Computer and Information Technology (CIT)*, pages 112–115. IEEE, 2016.
46. Yu-Bo Sheng and Lan Zhou. Distributed secure quantum machine learning. *Science Bulletin*, 62(14):1025–1029, 2017.
47. Ghanshyam Verma and Priyanka Tripathi. Scaling applications on cloud using GPGPU-trends and techniques. In *2016 Fifth International Conference on Eco-friendly Computing and Communication Systems (ICECCS)*, pages 89–93. IEEE, 2016.
48. Jun Wang, Alla Zelenyuk, Dan Imre, and Klaus Mueller. Big data management with incremental k-means trees–gpu-accelerated construction and visualization. In *Informatics*, volume 4, page 24. Multidisciplinary Digital Publishing Institute, 2017.
49. ShuHao WANG and GuiLu LONG. Big data and quantum computation. *Chinese science bulletin*, 60(5-6):499–508, 2015.
50. Zhenyu Wen, Pramod Bhatotia, Ruichuan Chen, Myungjin Lee, et al. Approxiot: Approximate analytics for edge computing. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 411–421. IEEE, 2018.
51. Wikipedia. Ai accelerator, 2020. [Online; accessed 25-Feb-2020].
52. Peter Wittek. *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.
53. Xuhong Zhang, Jun Wang, and Jiangling Yin. Sapprox: enabling efficient and accurate approximations on sub-datasets with distribution-aware online sampling. *Proceedings of the VLDB Endowment*, 10(3):109–120, 2016.
54. Baoxue Zhao, Jianlong Zhong, Bingsheng He, Qiong Luo, Wenbin Fang, and Naga K Govindaraju. GPU-accelerated cloud computing for data-intensive applications. In *Cloud Computing for Data-Intensive Applications*, pages 105–129. Springer, 2014.
55. Jianlong Zhong and Bingsheng He. Towards gpu-accelerated large-scale graph processing in the cloud. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, volume 1, pages 9–16. IEEE, 2013.

Multi-Label Ranking: Mining Multi-Label and Label Ranking Data

Lihi Dery

1 Introduction

Multi-label ranking (MLR) is the problem of predicting and ranking multiple labels for a single instance. MLR can be typically reduced to two sub-problems: the first is to rank labels for each instance, and the second is to place a threshold on the ranked list in order to bipartite the data into relevant and irrelevant labels. The ranking may contain ties, in the extreme case relevant labels have a tie on first place, and irrelevant labels have a tie on second place. The labels predicted for each instance are known as the instance's *labelset*.

The first studies of MLR originated from the investigation of text categorization problems, where each document may belong to several predefined topics simultaneously [69, 87, 113]. The topic labels may be ranked in order of importance [68]. While multi-label for text categorization continues to be an active field [90, 92], multi-label ranking has spread to many more domains. In bioinformatics, a gene can belong to multiple functional families [33]. In music, a tune can spark many emotions [124]. In medical diagnosis, an x-ray image can have multiple labels [8]. In social networks, people may belong to several interest groups [135] and in visual object recognition, objects can be ordered according to their relevance to the picture [18, 151].

There are six common challenges in MLR that either do not exist in single-label classification or are intensified in MLR settings.

- **High dimensionality in the output space.** High dimensionality in the *input space* (i.e., data with millions of instances) or in the *feature space* (i.e., data with thousands or millions of features) is also common in single-label classification,

L. Dery (✉)
Ariel University, Ariel, Israel
e-mail: lihid@ariel.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_23

although in MLR it might be harder to solve. However, high dimensionality in the *output space* is unique to MLR. The number of possible labelsets (i.e., label combinations) grows exponentially with the number of labels. This often leads to sparseness of available data and to class imbalance, as some labelsets may appear often, while others may be rare or may not appear in the training set at all. See an example in Fig. 1.

- **Label correlation.** This aspect is fundamental in MLR. If there are no relations between the labels, the problem can be split into multiple binary classification problems without loss of information. The relation between the labels is complex. For example, if two labels have a high concurrence, the model is supposed to somehow boost the prediction of one label, if the other is predicted. If two labels have a parallel relation, i.e., they do not concur, the model is expected to handle that as well. See an example of label concurrence in Fig. 2.

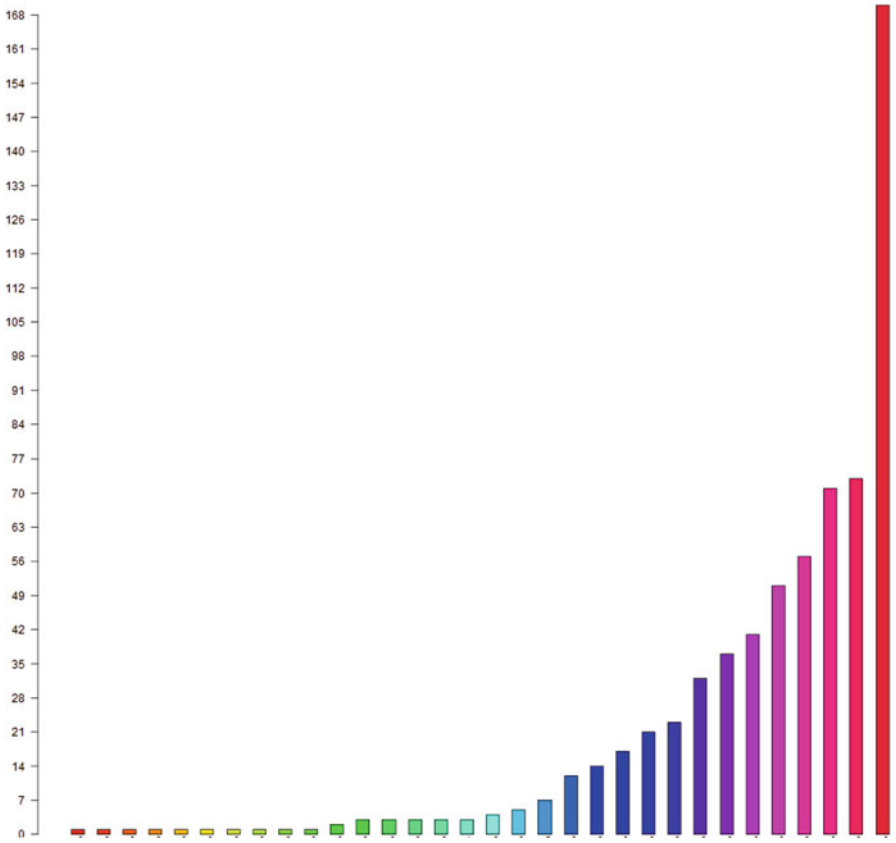


Fig. 1 High dimensionality in the output space. The number of instances (y-axis) with a given labelset (x-axis) in the Genebase dataset

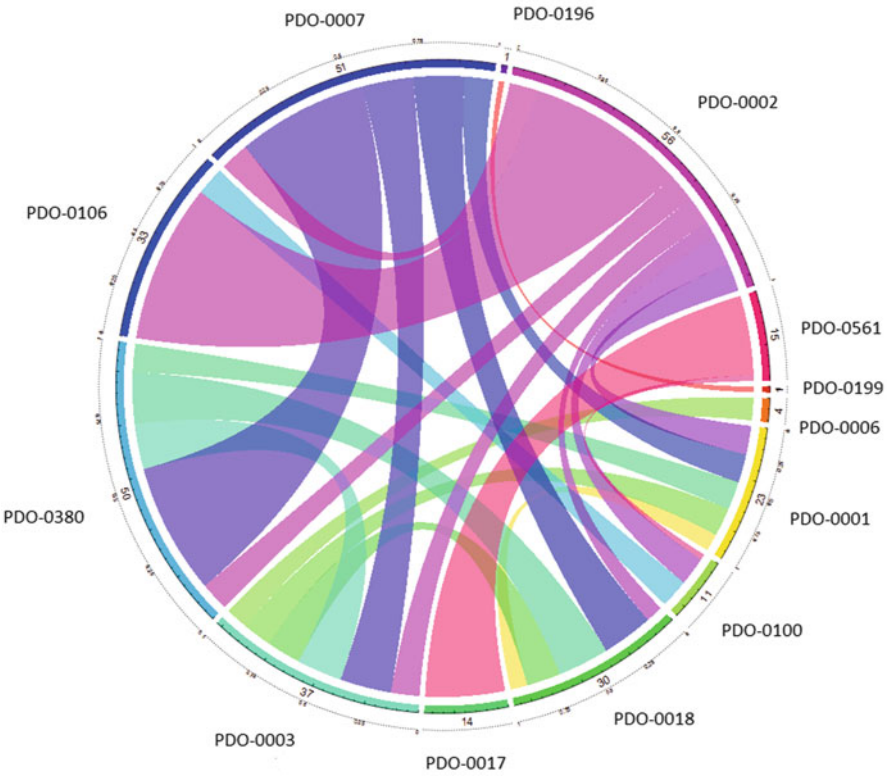


Fig. 2 Label correlation: a chord diagram [54] showing the concurrence of 13 labels in the Genebase dataset. The arcs represent label concurrence For example, the protein to the right of 12 o'clock (PDO-0196) co-occurs with only one other protein (PDO-0199) and that happens only once

- **Label imbalance.** The label distribution is highly skewed, most labels have only a handful of positive training instances and a few labels dominate with many training instances. See an example in Fig. 3.
- **Labelset size imbalance.** The labelset size of each instance is highly skewed [110]. Few instances have much more labels than average, while most instances have very few labels. See an example in Fig. 4.
- **Label importance.** Not all labels are equally important to the characterization of the instance. The label importance is explicitly known if the target class input labelset is ranked (i.e., if a ranked labels are provided as input in the training data). Otherwise, the label importance remains to be inferred.
- **Zero-shot labels and labelsets.** Some labels and labelsets never appear in the training set.

Figures 1, 2, 3, and 4 were created using the mldrGUI [22] with the Genebase dataset [39] as an example. The dataset contains the classification of proteins into

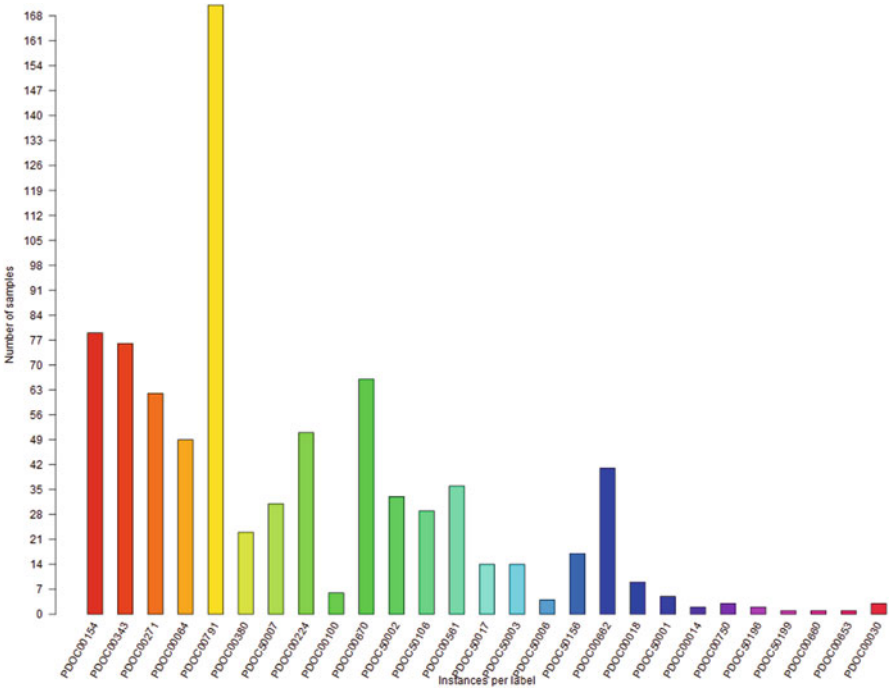


Fig. 3 Label imbalance. The number of instances (y-axis) with a given label (x-axis) in the Genebase dataset

families with similar function. Each instance is a protein; the attributes are the protein's motifs. The labels are the families the protein belongs to. This is a multi-label setting since each protein can belong to more than one family. In its current online version [128], the dataset contains 662 instances (the proteins) with 1213 attributes each. There are 27 possible labels (families) and 32 possible labelsets.

Previous literature reviews on multi-label methods [47, 48, 61, 161] and label ranking methods [131, 166], while excellent, are slightly outdated. Our contributions in this survey are three-fold. First, we refresh the definition of MLR [17] and define its two sub-tasks: multi-label and label ranking. Second, we suggest to re-categorize MLR methods, as they no longer fit into the traditional categories of transformation or adaptation. We thus suggest new categories such as deep learning multi-label methods, extreme multi-label methods and label ranking methods. Third, we focus on the last demi-decade which has not yet been surveyed.

The rest of this survey is organized as follows: we first define MLR and place it in context with other tasks (Sect. 2). Next, we survey recent developments in multi-label methods, with a special focus on deep learning methods and the emerging field of extreme multi-label classification (Sect. 3). We then move on to discuss label ranking (Sect. 4). We present up to date information about evaluation (Sect. 5) and conclude by offering a few research directions on open problems (Sect. 6).

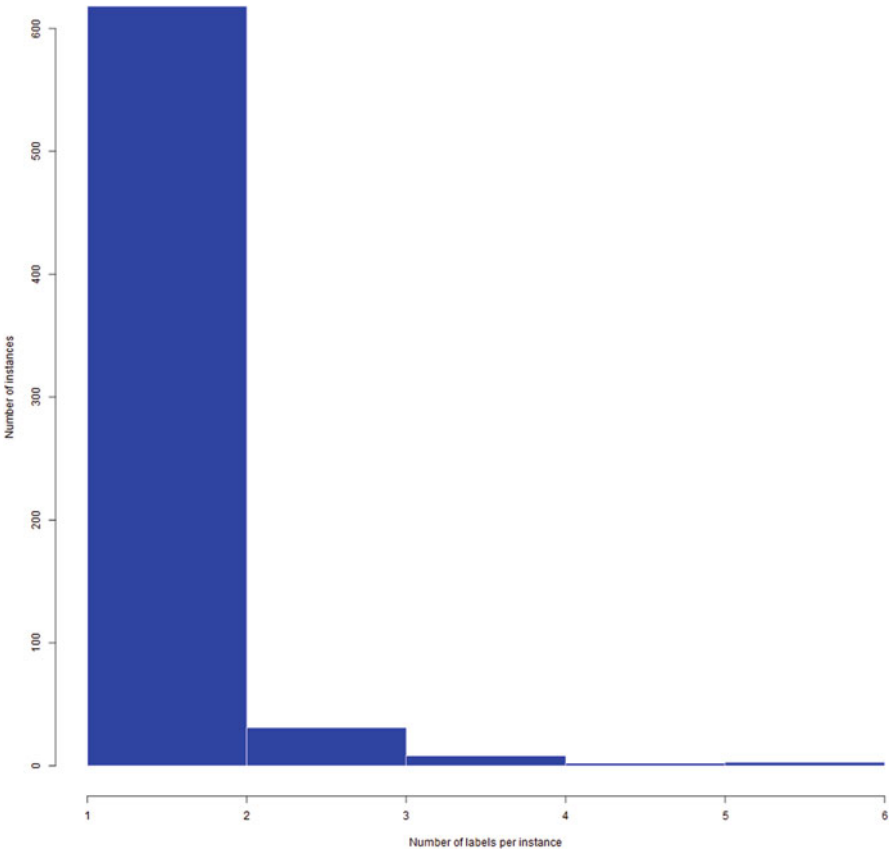


Fig. 4 Labelset size imbalance. The number of instances (y-axis) with a given number of labels (x-axis) in the Genebase dataset

2 Definition and Context

We begin with a definition of MLR. Next we place MLR in context by detailing which problems can be seen as sub-cases of MLR, and what MLR is a sub-problem of.

Definition 1 Multi-label ranking (MLR) An MLR task is characterized by $\mathbf{x} \in \mathcal{X}$ instances and $l \in L$ labels with the following properties:

1. $\mathcal{X}$ is finite and contains n instances.
2. L is finite and contains m labels.
3. An ordered labelset $Y = [l_1, \ldots, l_q]$ with $q \leq |L|$ labels contains a subset of the L possible labels. l_1 is the label with the highest rank and l_q is the label with the lowest rank.
4. The labelset size is exponential to the amount of labels: $Y \subseteq \mathbb{P}(L)$.

5. Ties in the labelset ranking are allowed, and in some cases all of the labels are tied in first place. A threshold $t \in 1, 2, .., q$ indicates the partitioning of Y into relevant and irrelevant labels. When $t = 1$ only the first label in Y is relevant. When $t = q$, or when the threshold is not mentioned, all labels in Y are relevant. When $1 \leq t \leq q$ the labelset is bipartite according to t.
6. The training dataset $\mathcal{D}$ consists of triplets $\{x_i, Y_i, t_i\}$
7. The goal is to find a mapping function: $h : \mathcal{X} \rightarrow Y$ for a given t.

In **multi-label ranking** problems the labelset is bipartite, i.e., the instance belongs to a ranked subset of the labels and does not belong to the rest of the set. For example, this image contains mainly oranges, apples, and bananas in this order, but no pears. Several problems are special cases of multi-label ranking: binary classification, multi-class classification, multi-label classification, and label ranking. In **binary classification**, an instance can belong to one of two possible classes, e.g., this image contains either an apple or an orange. In **multi-class** problems, an instance can belong to one out of multiple possible classes, e.g., this image contains either an apple, an orange or a banana. In **multi-label** problems, an instance can belong to many classes (labels), e.g., this image contains an apple and an orange but not a banana. In **label ranking** problems, an instance belongs to a ranked set of classes (labels), e.g., this image contains oranges, apples, and bananas, in this order.

Multi-label ranking tasks, as defined in Definition 1, refer to any problem whose *target class output* is a ranked list of labels and a threshold. Algorithms for solving these tasks are often divided into two sub-groups according to their *target class input*. When the input is a ranked set of labels, it is a label ranking task. When the input is just a set of labels, it is a multi-label task. Formally:

Definition 2 **A multi-label (ML)** task is an MLR with $1 \leq t \leq q$. The labelset is bipartite according to t, with labels $1 \leq t$ tied in first place and considered as relevant.

Definition 3 **A label ranking (LR)** task is an MLR with $t = q$ and no ties in Y.

Table 1 summarizes the differences between MLR and its sub-tasks according to a few parameters:

- Is $m > 2$?—There may be many labels available for each instance, or just two.
- Is $q > 1$?—It may be possible to assign many labels for each instance, or just one.

Table 1 Table of sub-cases of multi-label ranking

	$m > 2$	$q > 1$	ties in Y	$t = q$
Multi-label Ranking (MLR)	yes	yes	yes	yes
Label ranking (LR)	yes	yes	yes	no
Multi-label (ML)	yes	yes	no	yes
Multi-class (MC)	yes	no	no	yes
Binary classification	no	no	no	no

- Are there ties in Y?—Is the labelset completely ordered, or can ties between labels exist?
- Does $t = q$?—Are all labels in Y relevant, or does the threshold t partitions the labels into relevant and irrelevant labels?

A recent survey categorizes multi-label mining as a sub-problem in multi-target learning [133]. Related problems in the multi-target domain, which are not covered in this survey, include:

- **Multi variate regression** [12] and **dyadic prediction** where the goal is to predict a score for the fit between the instance and the label.
- **Hierarchical multi-label** [20] where there is explicit side information about the dependencies between the labels.
- **Multi-instance** [62] and **multi-instance multi-label** [168, 65] where the training data is composed of a bag of instances that are all assigned the same label or labels.
- **Multi-view** [165] is similar to multi-instance but the instances may have different feature spaces.

3 Multi-Label Algorithms

To date, surveys classify multi-label algorithms as either problem transformation techniques or algorithm adaptation techniques [21, 48, 161]. In the first, the problem is transformed into a simpler single-label classification task. In the second, an algorithm used for single-label classification is adapted to perform multi-label tasks. Sometimes ensembles techniques are added as a third class of problems and other times they are listed by their underlying base classifier's category (transformation or adaptation).

However, in the last decade algorithms that are specially designed for multi-label tasks have emerged. These algorithms either: try to maximize a specific evaluation measure (e.g., [98, 99, 143]), focus on a certain sub-task (e.g., feature selection [97, 118, 162]), or attempt to address specific multi-label challenges such as high dimensionality output space, label correlation (e.g., [53, 66]), imbalance, or zero-shot (e.g., [108]).

Moreover, deep learning algorithms designed specifically for multi-label tasks have been rapidly developing, exhibiting promising results. Indeed, a minority of these algorithms can be classified as algorithm transformation techniques (see, e.g., the method suggested in [151]), but more often, the algorithm is not adapted but rather specially tailored for the problem at hand.

As transformation and adaptation techniques have been exquisitely covered [9, 21, 48, 61, 158, 161] and as there is only a minor increase in research on them in comparison with other aspects of MLR, we lightly scan these foundations (Sects. 3.1 and 3.2), and then direct our focus on methods from the last demi-decade, specifically on deep learning (Sect. 3.3) and extreme multi-label methods (Sect. 3.4).

3.1 Problem Adaptation and Problem Transformation

Problem adaptation techniques were originally suggested for text categorization. However, the adaptations soon expanded beyond the text domain and into other scenarios as well. Some of the most noticed adaptations include: Expectation maximization (EM) [87], SVM [42, 69, 147, 145], k-NN [19, 31, 64, 117, 160], decision trees [4, 33], association rules [122], and genetic algorithms [50].

Problem transformation techniques [9] focus on transforming the problem into simpler sub-tasks. One classifier is created for each label or pair of labels. The classifiers are then trained separately, and their output is combined. The possible transformations are:

- **Binary Relevance.** The transformation into a binary classification task is known as binary relevance (BR) [158]. The first BR solutions [13, 49] did not consider label correlation. However, many correlation-enabling extensions to binary relevance have been proposed in the past decade. These correlations are classified into three sub-classes: first order correlation, pairwise correlation, or full correlation [161].
- **Multi-class.** The most known multi-class transformations are the Label Powerset methods that reduce the problem to a multi-class one by treating each individual labelset as an independent class label [13]. In both BR and multi-class transformations, there is a computational complexity problem, as the solutions do not scale well as the number of labels increase. Thus solutions that reduce the number of classifiers were suggested [86, 88].
- **Pairwise label comparisons.** Calibrated Label Ranking [46] transform the dataset into pairs of labels and thus train $k(k\ 1)/2$ binary classifiers. The output of the classifiers is combined into a ranking of the output labels, with the highest ranked labels considered as relevant. A fictional label can be used to automatically create a bi-partition of the labels into relevant and irrelevant ones [17].

3.2 Multi-Label Ensembles

The 2BR method [125] uses BR twice and employs stacking. It first learns a BR model, and then builds a second, meta-model that takes the output of the first model and includes an explicit coefficient for correlated labels. The PruDent method focuses on unnecessary label dependencies and error-propagation showing improved results over 2BR [5].

In Classifier chains (CC) [104], the first classifier is trained on the input attributes. The classifier's output is then added as a new input attribute, and a second classifier is trained, and so on. In this way the classifiers are chained, taking into account the possible label dependencies. In ensembles of classifier chains (ECC) [104], a set of CCs with different orders are trained and the outputs are aggregated.

Hierarchy Of Multi-label classifiERs (HOMER) [126] creates a tree of BR methods, where each leaf contains one label. To classify a new instance, HOMER begin at the root classifier and passes the instance to each child only if the parent predicted any of its labels. The union of the predicted labels by the leaves generates output for the given instance.

AdaBoost.MH [113] is the multi-label variation of the well-known AdaBoost algorithm [45]. AdaBoost.MH weighs the labels as well as the instances. Training instances and their corresponding labels that are hard to predict, get incrementally higher weights in following classifiers while instances and labels that are easy to classify get lower weights. This algorithm is designed to minimize the hamming loss. ADTBoost.MH [35] which uses ADTTrees is an extension of AdaBoost.MH.

Random k-Labelsets (RAkEL) [129] select a number of random k-labelsets and learn a Label Powerset classifier for each of them. These are then aggregated. Enhancements over RakEL include RakEL++ [109] and RAkELLd [127].

An experimental study on most of the above multi-label ensembles suggests that ECC, followed by RakEL, exhibit the best overall performance for all of the examined metrics [91].

A few notable ensemble methods were published in the last 3 years. ML-FOREST [141] builds on Random Forest and ML-TSVM [27] on SVM. PRAkEL, a cost-sensitive extension of RakEL that considers the evaluation criteria and is sensitive to the cost of misclassifying an instance [143]. fRAkEL speeds up RakEL by shrinking the samples with irrelevant labels [146]. The TSEN ensemble [164] is based on three-way-decisions [154]. The MULE ensemble [95] relies on a heterogeneous ensemble that is composed of different base models. The assumption is that different labels can be approximated better by different types of models. MULE incorporates a statistical test to combine the base models. Most of these methods compare their performance to earlier methods, mainly to variations of 2BR, ECC, AdaBoost.MH, and RakEL. However, an evaluation of these methods in comparison to one another is still missing.

3.3 Deep Learning Methods

Deep learning uses multiple layers to represent the abstractions of data and to automatically discover useful features [79, 114]. Deep learning can cope with large amounts of input features, eliminating the need for feature selection methods. The learnt feature representations are often accurate for other unseen data as well. While this quality, known as transfer learning [94], is not unique to deep learning, in the case of multiple labels it can provide a solution to the label and labelset imbalance problems. This also means that parameters for a new model (such as number of layers and number of nodes) can be learnt from previous successful models instead of by trial and error. Deep learning models are devised using different architectures [100]. Convolutional Neural Network (CNN) models [34, 80] are often used for image processing. Recurrent neural network (RNN) models [78] and their variant

Long Short-Term Memory (LSTM) [43, 119] are often used with text and speech. Generative adversarial networks (GANs) [52] and Restricted Boltzmann Machine (RBM) [1, 121] have been used for unsupervised multi-label learning tasks but we herein focus on a supervised MLR setting as defined in Definition 1.

Deep learning for multi-label tasks is a growing field, with new papers appearing frequently. We survey some of the recent developments in two main multi-label tasks, which belong to the media domain: image annotation and text annotation.

3.3.1 Image Annotation

The growing interest in deep learning for multi-label image annotation is partly driven by new publicly accessible large-scale datasets with quality labels. A few of the most notable ones include:

- **The Visual Genome dataset [74].** Contains over 108K images with an average of 35 labeled objects, 26 attributes, and 21 pairwise relationships between objects.
- **The ChestX-ray14 dataset [136].** Contains over 112k chest X-rays from over 30k patients, labeled with up to 14 pathologies or "No Finding".
- **The MS COCO dataset [83].** Contains 328k images with 2.5 million labeled objects, out of a set of 91 labels.
- **The NUS-WIDE dataset [32].** Contains almost 270k Flickr images and their associated labels, with a total of 5018 unique labels.

Recent reviews of deep learning for medical images highlight the vast amount of emerging research in this field [44, 70, 112]. On the chest X-rays dataset, various CNN based methods have been suggested. One approach is to transform the problem into multiple single-label classification problems, to which a CNN architecture is applied [51]. Another suggestion is Hypotheses-CNNPooling (HCP), where a number of object (i.e., image) segment hypotheses are taken as the inputs, then a shared CNN is connected with each hypothesis, and finally the CNN output results from different hypotheses are aggregated with max pooling to produce the multi-label predictions [138]. The CNN-RNN model uses an underlying RNN model [89] to capture the high order label dependencies. Then, CNN and RNN are combined into one framework to exploit the label dependencies at the global level [134]. Other models exist (e.g., [40, 55, 56, 57, 103, 116]), as well as a cascade ensemble [76]. Though multi-instance methods are out of our scope, we note that a multi-instance method that integrates the images with other information about the patients has recently been reported to enhance performance [8].

Various studies have been conducted on non-medical images as well. For a recent review see Voulodimos et al. [132]. Some of these models focus on learning the label correlations. A model that learns image-dependent conditional structures [82] has been proposed. A Spatial Regularization Network (SRN) that captures the spatial correlation between labels as well as the semantic correlation has been suggested [169]. A feature attention network (FAN) focuses on more important features and

learn the correlations among convolutional features [150]. The Regional Latent Semantic Dependencies (RLSD) model [157] specializes in predicting small objects (alongside prediction of large objects) by first extracting convolutional features, which are further sent to an RPN-like (Regional Proposal Network) localization layer. The layer is designed to localize the regions in an image that may contain multiple semantically dependent labels. These regions are encoded with a fully connected neural network and further sent to an RNN, which captures the latent semantic dependencies at the regional level. The RNN unit sequentially outputs a multi-class prediction, based on the outputs of the localization layer and the outputs of previous recurrent neurons. Finally, a max-pooling operation is carried out to fuse all the regional outputs as the final prediction. Once again, multi-instance methods can enhance performance [85].

Comparisons of cutting-edge image annotation methods are still lacking. A precedent study compared the performance of ten foreground deep learning multi-label APIs on the Visual Genome dataset [75]. The APIs were evaluated using various metrics. In addition, a semantic similarity metric was used allowing for words with similar meaning to be classified as correct predictions. For example, "bicycle" and "bike" were both classified as correct for an image of a pedal driven two-wheeler. The study shows that different APIs excel under different evaluation metrics. Regretfully, the underlying algorithm of each API is not always publicly available.

3.3.2 Text Annotation

One of the earliest models to employ deep learning for text classification was BP-MLL [159]. It formulated multi-label classification problems as a neural network with multiple output nodes, one for each label. It was later suggested [93] to replace the pairwise ranking loss in the model with a cross-entropy loss instead. However, these models do not consider label dependencies. A CNN model that has a final hidden layer which considers label co-occurrence weights was suggested next [77]. The model was analyzed on a small dataset with 103 labels. In the CNN-RNN model [26], the RNN is set to deal with label co-occurrence. The C2AE algorithm employs a DNN-based label embedding framework and performs joint feature and label embedding [155].

The Seq2seq model [152] uses a LSTM to generate labels sequentially and predicts the next label-based on its previously predicted labels. The $LSTM^2$ model [149] utilizes LSTM twice. The algorithm first builds a representation of the documents in the training set. This is done with a LSTM network that considers word sequences. For each document in the test set, the algorithm search for the most similar documents in the training set and retrieves their labels. The labels are represented as a semantic tree that is trained with dependency parsing. This tree can capture the correlations between labels. Based on the document representation, another LSTM is utilized to rank the document labels.

Recently, text categorization has also been employed in the context of X-ray images. For example, it has been pointed out that China's chest X-ray reports focus more on characterization than on labeling the possible diseases. A recent study uses LSTM to read the X-ray reports and output labels of pathologies [148]. Another study combines the reports and the instances (a multi-instance model) for the same purpose [137].

3.4 Extreme Multi-Label Classification

Extreme multi-label (XML) problems, also known as large-scale multi-label problems, refer to problems with an extremely large set of labels, usually in the millions. Due to the huge amount of labels, the zero-shot problem, as well as the label and labelset imbalance problems (see Sect. 1), are intensified. Deep learning models work well here since they consider both the relatedness of the representations and the context information.

Currently, all of these problems focus on text classification or on traditional recommender system problems that have been reformulated as XML problems [120, 130]. The instances are of one of the following kinds:

- **Text.** Assigning categories (the labels) to a Wikipedia page out of the million categories (labels) available, recommending bid phrases (the labels) to an advertiser with a given ad landing page [2], assigning tags (the labels) to an image. In these cases, the instance, be it a web page or an image, is represented by a bag of words.
- **Item.** Assigning a number of categories (the labels) to an Amazon product item. The instance is represented by item features.
- **User.** Recommending YouTube videos (the labels) to a user [140]. The instance is represented by user features.

We note that these tasks often require a ranked output, and that a ranked input might be available. However, to the best of our knowledge, extreme label ranking which explicitly considers ranked inputs (as opposed to extreme multi-label) has not yet been explored.

PDSparse [156] and DiSMEC [7] tackle the sparseness (high dimensionality) by training one XML model per label. We thus consider them as XML problem transformation methods.

MLRF [2] designs a multi-label random forest. To cope with the high dimensionality problem, they assume label independence during the ensemble construction, but they do consider correlations during prediction. The FastXML model [102] provides a node partitioning formulation that improves MLRF results in settings with millions of labels. XML-CNN [84] uses a CNN model for the actual classification.

The SwiftXML model [101] learns label correlation using a word2vec embedding. This allows the discovery of similar labels as these labels are classified closely in the embedded vector. An example given by the authors is that although the labels

of the Wikipedia pages of Einstein and Newton are very different, the SwiftXML label embedding will learn that the two are similar, and thus will be able to consider labels given to Einstein's page, also for Newton's page. Authors report to have outperformed SLEEC [10] which also employs label embedding and is considered state-of-the-art. A recent model [163] tackles both feature and label spaces using a non-linear embedding based on a graph structure.

4 Label Ranking Algorithms

Numerous label ranking algorithms were suggested in the literature. One approach is based on turning the problem into several binary classification problems and then combining them into output rankings [29, 38, 58, 60, 67]. Another common approach is based on modifying existing probabilistic algorithms to directly support label ranking. Some main examples are: naive Bayes models [3], k-nearest neighbor models [14], and decision tree models such as Label Ranking Trees (LRT) [30] and Entropy Based Ranking Trees (ERT) [36].

A few other stand-alone ideas are available as well. RPC (Ranking by Pairwise Comparison) [67] learns pairwise preferences from which a ranking is derived. Instance-Based Logistic Regression (IBLR) combines instance-based learning and logistic regression [30]. Under this approach, the label statistics of neighboring instances are regarded as features by the logistic regression classifier. A rule based approach learns a reduction technique and provides a mapping in the form of logical rules [59]. A recent work [72] adapts ideas from structured output prediction. They cast the label ranking problem into the structured prediction framework and propose embeddings dedicated to ranking representation. For each embedding they propose a solution to the pre-image problem. This latter suggestion is a harbinger for a bridge between label ranking and structured image prediction.

Surveys on label ranking algorithms [131, 166] capture some of the earlier methods.

4.1 Label Ranking Ensembles

To the best of our knowledge, only a few papers thus far have investigated the use of ensembles for label ranking [6, 111, 139, 167]. The ensembles proposed in these papers differ in: (1) the base label ranking algorithm used, (2) the method used to sample the data to train each of the simple models (if all models are trained with the exact same data they will output the exact same results, and then there is no need for an ensemble), and (3) the aggregation method used to combine the results of the simple models.

As the base label ranking algorithm [6] used Label Ranking Trees (LRT) [30], [111] used Ranking Trees (RT) and Entropy Ranking Trees (ERT) [36], [167]

Table 2 Label ranking ensemble frameworks

	Aledo et al. [6]	Sa et al. [111]	Zhou and Qiu [167]	Werbin et al. [139]
Base algorithm	LRT	ERT, RT	TLAC	LRT, RPC
Data sampling	Bagging	Random Forest	Random Forest	Bagging
Aggregation	Modal ranking	Borda	Borda	Voting Rule Selector

developed their own method named Top Label As Class (TLAC) and [139] used both LRT and Ranking by Pairwise Comparison (RPC) [67].

To select the training data for each simple classifier, [6] and [139] used a technique known as Bootstrap aggregation or Bagging [15]: they created b different bags by selecting a subset of the dataset's instances with replacement. The other two papers [111, 167] suggested modifications to the well-known Random Forest ensemble model [16]. As for the aggregation method used, three studies used a voting rule (either Borda or Modal Ranking). The last study [139] presents VRS (Voting Rule Selector), a meta-model that automatically learns the best voting rule to be used. There four works are summarized in Table 2.

AdaBoost.MR is the label ranking variation of the well-known AdaBoost algorithm [45]. The algorithm performs pairwise comparisons between labels. Training instances and their corresponding label pairs that are hard to predict, get incrementally higher weights in following classifiers while instances and label pairs that are easy to classify get lower weights. The algorithm is designed to minimize the ranking loss. Another boosting-based approach suggests learning a linear utility function for each label, from which the ranking is deduced [37]. This approach is more general, as it allows the input to be any sort of preference graph over the labels. A ranking function assigns a utility score to each label. Then, relevant pairwise comparisons, as deduced from the graph, are performed. Again, the label weights are updated and additional weight is given to each wrongly ranked pair. As these approaches rely on pairwise comparisons, they do not scale well in the case of many labels.

5 Evaluation

Evaluating the performance of multi-label algorithms is difficult. For example, it may be impossible to decide which mistake of the following two cases is more serious: one instance with three incorrect labels vs. three instances each with one incorrect label. Therefore, a number of performance measures focusing on different aspects have been proposed. Schapire and Singer [113] initially suggested five metrics: Hamming loss, ranking loss, one-error, average precision. The popular micro-F1 and macro-F1 were suggested by Tsoumakas et al. [127]. Instance-F1 and AUC measures were discussed by Koyejo et al. [73]. A summary of the 11 most common measures is provided Wu et al. [142]. For label ranking, the most popular

measure is Kendall-tau. The hamming distance is also often used when permutations represent matching of bipartite graphs.

5.1 Dataset Repositories

The Mulan repository [128] contains 27 multi-label datasets on various subjects. MEKA [105] contains datasets in ARFF format, suitable for Weka. An R package automates the use of these datasets [23]. The KDIS research group offers a repository of various datasets obtained from different sources.[1]

The Extreme Classification Repository stores 15 extreme multi-label text datasets, as well as code for various algorithms.[2] LSHTC series challenge also stores extreme multi-label text datasets [96]. The SNAP library [81] contains social and information networks, some of which can (and were) utilized for extreme multi-label tasks (e.g., Amazon products). Lastly, it is possible to generate simulated data via a multi-label data generator [123].

As for datasets with a ranked target class as input, five real-world label ranking datasets can be found in [106]. The 16 semi-synthetic label ranking datasets used in [28, 30] are stored on the webpage of one of the authors.[3]

5.2 Stratification of Multi-Label Data

Estimating the accuracy of a model is traditionally done by splitting the data to training, test, and sometimes also validation subsets. Different techniques are available, such as cross-validation, holdout, and bootstrap [41, 107]. Random sampling works well when the labels have enough representation in the data. When this is not the case, a stratification approach assures that all subsets contain approximately the same proportions of labels as the original dataset. For single-label classification tasks, stratification has been shown to outperform cross-validation with random sampling [71].

For multi-label data, stratification is even more important, as some datasets may have very few patterns representing them and random sampling might place them all in either the training or the test data partitions.

Stratification for multi-label data can either consider the distinct labelsets available in the data or consider each label separately. Since the number of labelsets grows exponentially to the number of labels, when the number of labels is large there might be only one instance for each example in the labelset, or even no examples

[1] http://www.uco.es/kdis/mllresources/.

[2] http://manikvarma.org/downloads/XC/XMLRepository.html#Prabhu14.

[3] https://cs.uni-paderborn.de/?id=63912.

at all. For this reason, a method that considers each label separately was suggested by Sechidis et al. [115]. Stratifying the data in this method is a slow procedure, but nonetheless, it improves the classifier performance.

6 Research Directions and Open Problems

There are still many paths to explore for multi-label ranking. We outline a few of them here.

First, in many real-world scenarios, new labels emerge over time. For example, the label "valentine2030" will only emerge on year 2030. Multi-Label learning with emerging new labels has just begun to be considered, and we are aware of only one pioneering work on the subject [170]. We have seen herein that recommender system problems can be reformulated as multi-label ones. Perhaps it is time to reconsider the other direction as well [153]. The "cold-start" problem is fundamental in recommender systems. Perhaps their solutions can be adapted to multi-label ranking tasks. Moreover, evaluation measures need to be fine-tuned to this setting of emerging labels [108, 144].

Second, a recent study [11] compared recommender system and multi-label classification techniques concluding that AdaBoost with CC chains and BR with multi-label random forest outperform the best recommender system methods in a given cross-selling setting. However, state-of-the-art multi-label deep learning methods and extreme multi-label methods should be able to do even better but have not been considered in the above study. Moreover, it is interesting to reconsider other recommender and ranking scenarios and reformulate them as MLR tasks as well.

Third, for single-label problems, a set of complexity measures that calculate the overlap and separability of classes has been defined [63]. To the best of our knowledge, one characterization metric, called TCS (Theoretical Complexity Score), exists for multi-label tasks [24] but no complexity measures exist for label ranking, where the emphasis is on the correct order of labels. In a related context, SCUMBLE (Score of ConcUrrence among iMBalanced LabEls) is a new metrics that address label co-occurrence in multi-label datasets [25]. It can and should be used to gain better understanding of the available data.

Fourth, we have reviewed state-of-the-art deep learning algorithms for image annotation and separate algorithms for text classification. One cannot but wonder if these fields can inspire one another and furthermore, if a meta-method for both tasks can be developed.

Lastly, as explained in Sect. 3.4, extreme label ranking problems where the target input space is ordered have yet to be explored.

References

1. David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.
2. Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24. ACM, 2013.
3. Artur Aiguzhinov, Carlos Soares, and Ana Paula Serra. A similarity-based adaptation of naive bayes for label ranking: Application to the metalearning problem of algorithm recommendation. In *International Conference on Discovery Science*, pages 16–26. Springer, 2010.
4. Reem Al-Otaibi, Meelis Kull, and Peter Flach. Lacova: A tree-based multi-label classifier using label covariance as splitting criterion. In *2014 13th International Conference on Machine Learning and Applications*, pages 74–79. IEEE, 2014.
5. Abdulaziz Alali and Miroslav Kubat. Prudent: A pruned and confident stacking approach for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2480–2493, 2015.
6. Juan A Aledo, José A Gámez, and David Molina. Tackling the supervised label ranking problem by bagging weak learners. *Information Fusion*, 35:38–50, 2017.
7. Rohit Babbar and Bernhard Schölkopf. DiSMEC: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 721–729. ACM, 2017.
8. Ivo M Baltruschat, Hannes Nickisch, Michael Grass, Tobias Knopp, and Axel Saalbach. Comparison of deep learning approaches for multi-label chest x-ray classification. *Scientific reports*, 9(1):6381, 2019.
9. Priyadarshini Barot and Mahesh Panchal. Review on various problem transformation methods for classifying multi-label data. *International Journal of Data Mining And Emerging Technologies*, 4(2):45–52, 2014.
10. Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*, pages 730–738, 2015.
11. Matthias Bogaert, Justine Lootens, Dirk Van den Poel, and Michel Ballings. Evaluating multi-label classifiers and recommender systems in the financial service sector. *European Journal of Operational Research*, 2019.
12. Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larrañaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015.
13. Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.
14. Pavel B Brazdil, Carlos Soares, and Joaquim Pinto Da Costa. Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277, 2003.
15. Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
16. Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
17. Klaus Brinker, Johannes Fürnkranz, and Eyke Hüllermeier. A unified model for multilabel classification and ranking. In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29–September 1, 2006, Riva del Garda, Italy*, pages 489–493. IOS Press, 2006.
18. Serhat S Bucak, Pavan Kumar Mallapragada, Rong Jin, and Anil K Jain. Efficient multi-label ranking for multi-class learning: application to object recognition. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2098–2105. IEEE, 2009.

19. Jorge Calvo-Zaragoza, Jose J Valero-Mas, and Juan R Rico-Juan. Improving kNN multi-label classification in prototype selection scenarios using class proposals. *Pattern Recognition*, 48(5):1608–1622, 2015.
20. Ricardo Cerri, Rodrigo C Barros, and André CPLF de Carvalho. Hierarchical multi-label classification for protein function prediction: A local approach based on neural networks. In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 337–343. IEEE, 2011.
21. David Charte, Francisco Charte, Salvador García, and Francisco Herrera. A snapshot on non-standard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations. *Progress in Artificial Intelligence*, 8(1):1–14, Apr 2019.
22. Francisco Charte and David Charte. Working with multilabel datasets in R: The mldr package. *The R Journal*, 7(2):149–162, December 2015.
23. Francisco Charte, David Charte, Antonio Rivera, María José del Jesus, and Francisco Herrera. R ultimate multilabel dataset repository. In Francisco Martínez-Álvarez, Alicia Troncoso, Héctor Quintián, and Emilio Corchado, editors, *Hybrid Artificial Intelligent Systems*, pages 487–499, Cham, 2016. Springer International Publishing.
24. Francisco Charte, Antonio Rivera, María José del Jesus, and Francisco Herrera. On the impact of dataset complexity and sampling strategy in multilabel classifiers performance. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 500–511. Springer, 2016.
25. Francisco Charte, Antonio J. Rivera, María J. del Jesus, and Francisco Herrera. Dealing with difficult minority labels in imbalanced multilabel data sets. *Neurocomputing*, 326–327:39–53, 2019.
26. Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2377–2383. IEEE, 2017.
27. Wei-Jie Chen, Yuan-Hai Shao, Chun-Na Li, and Nai-Yang Deng. MLTSVM: a novel twin support vector machine to multi-label learning. *Pattern Recognition*, 52:61–74, 2016.
28. Weiwei Cheng, Krzysztof Dembczyński, and Eyke Hüllermeier. Label ranking methods based on the Plackett-Luce model. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 215–222, Haifa, Israel, June 2010. Omnipress.
29. Weiwei Cheng, Sascha Henzgen, and Eyke Hüllermeier. Labelwise versus pairwise decomposition in label ranking. In *LWA*, pages 129–136, 2013.
30. Weiwei Cheng, Jens Hühn, and Eyke Hüllermeier. Decision tree and instance-based learning for label ranking. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 161–168. ACM, 2009.
31. Tsung-Hsien Chiang, Hung-Yi Lo, and Shou-De Lin. A ranking-based kNN approach for multi-label classification. In *Asian Conference on Machine Learning*, pages 81–96, 2012.
32. Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of Singapore. In *Proceedings of the ACM international conference on image and video retrieval*, page 48. ACM, 2009.
33. Amanda Clare and Ross D. King. Knowledge discovery in multi-label phenotype data. In Luc De Raedt and Arno Siebes, editors, *Principles of Data Mining and Knowledge Discovery*, pages 42–53, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
34. Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.
35. Francesco De Comité, Rémi Gilleron, and Marc Tommasi. Learning multi-label alternating decision trees from texts and data. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 35–49. Springer, 2003.

36. Cláudio Rebelo de Sá, Carla Rebelo, Carlos Soares, and Arno Knobbe. Distance-based decision tree algorithms for label ranking. In *Portuguese Conference on Artificial Intelligence*, pages 525–534. Springer, 2015.
37. Ofer Dekel, Yoram Singer, and Christopher D Manning. Log-linear models for label ranking. In *Advances in neural information processing systems*, pages 497–504, 2004.
38. Sébastien Destercke, Marie-Hélène Masson, and Michael Poss. Cautious label ranking with label-wise decomposition. *European Journal of Operational Research*, 246(3):927–935, 2015.
39. Sotiris Diplaris, Grigorios Tsoumakas, Pericles A Mitkas, and Ioannis Vlahavas. Protein classification with multiple algorithms. In *Panhellenic Conference on Informatics*, pages 448–456. Springer, 2005.
40. Yuxi Dong, Yuchao Pan, Jun Zhang, and Wei Xu. Learning to read chest X-ray images from 16000+ examples using CNN. In *Proceedings of the Second IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies*, pages 51–57. IEEE Press, 2017.
41. Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.
42. André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687, 2002.
43. Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.
44. Yeli Feng, Hui Seong Teh, and Yiyu Cai. Deep learning for chest radiology: A review. *Current Radiology Reports*, 7(8):24, 2019.
45. Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
46. Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153, 2008.
47. Eva Gibaja and Sebastián Ventura. Multi-label learning: a review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(6):411–444, 2014.
48. Eva Gibaja and Sebastián Ventura. A tutorial on multilabel learning. *ACM Computing Surveys (CSUR)*, 47(3):52, 2015.
49. Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 22–30. Springer, 2004.
50. Eduardo Corrêa Gonçalves, Alex A Freitas, and Alexandre Plastino. A survey of genetic algorithms for multi-label classification. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2018.
51. Yunchao Gong, Yangqing Jia, Thomas Leung, Alexander Toshev, and Sergey Ioffe. Deep convolutional ranking for multilabel image annotation. In *International Conference on Learning Representations (ICLR)*, 2014.
52. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
53. Quanquan Gu, Zhenhui Li, and Jiawei Han. Correlated multi-label feature selection. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1087–1096. ACM, 2011.
54. Zuguang Gu, Lei Gu, Roland Eils, Matthias Schlesner, and Benedikt Brors. circlize implements and enhances circular visualization in R. *Bioinformatics*, 30(19):2811–2812, 2014.
55. Qingji Guan and Yaping Huang. Multi-label chest x-ray image classification via category-wise residual attention learning. *Pattern Recognition Letters*, 2018.
56. Qingji Guan, Yaping Huang, Zhun Zhong, Zhedong Zheng, Liang Zheng, and Yi Yang. Diagnose like a radiologist: Attention guided convolutional neural network for thorax disease classification. *arXiv preprint arXiv:1801.09927*, 2018.

57. Sebastian Guendel, Sasa Grbic, Bogdan Georgescu, Siqi Liu, Andreas Maier, and Dorin Comaniciu. Learning to recognize abnormalities in chest x-rays with location-aware dense networks. In *Iberoamerican Congress on Pattern Recognition*, pages 757–765. Springer, 2018.
58. Massimo Gurrieri, Philippe Fortemps, and Xavier Siebert. Alternative decomposition techniques for label ranking. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 464–474. Springer, 2014.
59. Massimo Gurrieri, Xavier Siebert, Philippe Fortemps, Salvatore Greco, and Roman Słowiński. Label ranking: A new rule-based label ranking method. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 613–623. Springer, 2012.
60. Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification for multiclass classification and ranking. In *Advances in neural information processing systems*, pages 809–816, 2003.
61. Francisco Herrera, Francisco Charte, Antonio J Rivera, and María J Del Jesus. Multilabel classification. In *Multilabel Classification*, pages 17–31. Springer, 2016.
62. Francisco Herrera, Sebastián Ventura, Rafael Bello, Chris Cornelis, Amelia Zafra, Dánel Sánchez-Tarragó, and Sarah Vluymans. Multiple instance learning. In *Multiple instance learning*, pages 17–33. Springer, 2016.
63. Tin Kam Ho and Mitra Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 24(3):289–300, 2002.
64. Jun Huang, Guorong Li, Shuhui Wang, Zhe Xue, and Qingming Huang. Multi-label classification by exploiting local positive and negative pairwise label correlation. *Neurocomputing*, 257:164–174, 2017.
65. Sheng-Jun Huang, Wei Gao, and Zhi-Hua Zhou. Fast multi-instance multi-label learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.
66. Sheng-Jun Huang and Zhi-Hua Zhou. Multi-label learning by exploiting label correlations locally. In *Twenty-sixth AAAI conference on artificial intelligence*, 2012.
67. Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16–17):1897–1916, 2008.
68. George Sakkas Grigorios Tsoumakas Ioannou, Marios and Ioannis Vlahavas. Obtaining bipartitions from score vectors for multi-label classification. In *In 2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, pages 409–416, 2010.
69. Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.
70. Kenji Karako, Yu Chen, and Wei Tang. On medical application of neural networks trained with various types of data. *Bioscience trends*, 12(6):553–559, 2018.
71. Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, volume 14, pages 1137–1145. Montreal, Canada, 1995.
72. Anna Korba, Alexandre Garcia, and Florence d'Alché Buc. A structured prediction approach for label ranking. In *Advances in Neural Information Processing Systems*, pages 8994–9004, 2018.
73. Oluwasanmi O Koyejo, Nagarajan Natarajan, Pradeep K Ravikumar, and Inderjit S Dhillon. Consistent multilabel classification. In *Advances in Neural Information Processing Systems*, pages 3321–3329, 2015.
74. Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.
75. Adam Kubany, Shimon Ben Ishay, Ruben-sacha Ohayon, Armin Shmilovici, Lior Rokach, and Tomer Doitshman. Semantic comparison of state-of-the-art deep learning methods for image multi-label classification. *arXiv preprint arXiv:1903.09190*, 2019.

76. Pulkit Kumar, Monika Grewal, and Muktabh Mayank Srivastava. Boosted cascaded convnets for multilabel classification of thoracic diseases in chest radiographs. In Aurélio Campilho, Fakhri Karray, and Bart ter Haar Romeny, editors, *Image Analysis and Recognition*, pages 546–552, Cham, 2018. Springer International Publishing.
77. Gakuto Kurata, Bing Xiang, and Bowen Zhou. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526, San Diego, California, June 2016. Association for Computational Linguistics.
78. Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.
79. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
80. Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE, 2010.
81. Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.
82. Qiang Li, Maoying Qiao, Wei Bian, and Dacheng Tao. Conditional graphical lasso for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2977–2986, 2016.
83. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
84. Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM, 2017.
85. Yan Luo, Ming Jiang, and Qi Zhao. Visual attention in multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.
86. Gjorgji Madjarov, Dejan Gjorgjevikj, and Sašo Džeroski. Two stage architecture for multi-label learning. *Pattern Recognition*, 45(3):1019–1034, 2012.
87. Andrew McCallum. Multi-label text classification with a mixture model trained by EM. In *AAAI workshop on Text Learning*, pages 1–7, 1999.
88. Eneldo Loza Mencia and Johannes Fürnkranz. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer, 2008.
89. Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.
90. Marcin Michał Mirończuk and Jarosław Protasiewicz. A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications*, 106:36–54, 2018.
91. Jose M. Moyano, Eva L. Gibaja, Krzysztof J. Cios, and Sebastián Ventura. Review of ensembles of multi-label classifiers: Models, experimental study and prospects. *Information Fusion*, 44:33–45, 2018.
92. Ghulam Mujtaba, Liyana Shuib, Norisma Idris, Wai Lam Hoo, Ram Gopal Raj, Kamran Khowaja, Khairunisa Shaikh, and Henry Friday Nweke. Clinical text classification research trends: Systematic literature review and open issues. *Expert Systems with Applications*, 116:494–520, 2019.
93. Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. Large-scale multi-label text classification—revisiting neural networks. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2014.

94. Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.
95. Yannis Papanikolaou, Grigorios Tsoumakas, Manos Laliotis, Nikos Markantonatos, and Ioannis Vlahavas. Large-scale online semantic indexing of biomedical articles via an ensemble of multi-label classification models. *Journal of biomedical semantics*, 8(1):43, 2017.
96. Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artières, George Paliouras, Éric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Gallinari. LSHTC: A benchmark for large-scale text classification. *CoRR*, abs/1503.08581, 2015.
97. Rafael B Pereira, Alexandre Plastino, Bianca Zadrozny, and Luiz HC Merschmann. Categorizing feature selection methods for multi-label classification. *Artificial Intelligence Review*, 49(1):57–78, 2018.
98. James Petterson and Tibério S Caetano. Submodular multi-label learning. In *Advances in Neural Information Processing Systems*, pages 1512–1520, 2011.
99. Ignazio Pillai, Giorgio Fumera, and Fabio Roli. Designing multi-label classifiers that maximize f measures: State of the art. *Pattern Recognition*, 61:394–404, 2017.
100. Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and SS Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):92, 2019.
101. Yashoteja Prabhu, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 441–449. ACM, 2018.
102. Yashoteja Prabhu and Manik Varma. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272. ACM, 2014.
103. Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. CheXNET: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*, 2017.
104. Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333, 2011.
105. Jesse Read, Peter Reutemann, Bernhard Pfahringer, and Geoff Holmes. MEKA: A multi-label/multi-target extension to Weka. *Journal of Machine Learning Research*, 17(21):1–5, 2016.
106. Claudio Rebelo de Sa. Label ranking datasets. http://dx.doi.org/10.17632/3mv94c8jpc.2, 2018. Accessed: 2019-10-30.
107. Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of database systems*, pages 532–538, 2009.
108. Anthony Rios and Ramakanth Kavuluru. Few-shot and zero-shot multi-label learning for structured label spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2018, page 3132. NIH Public Access, 2018.
109. Lior Rokach, Alon Schclar, and Ehud Itach. Ensemble methods for multi-label classification. *Expert Systems with Applications*, 41(16):7507–7523, 2014.
110. Timothy N Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. Statistical topic models for multi-label document classification. *Machine learning*, 88(1–2):157–208, 2012.
111. Cláudio Rebelo Sá, Carlos Soares, Arno Knobbe, and Paulo Cortez. Label ranking forests. *Expert Systems*, 34(1), 2017.
112. Berkman Sahiner, Aria Pezeshk, Lubomir M Hadjiiski, Xiaosong Wang, Karen Drukker, Kenny H Cha, Ronald M Summers, and Maryellen L Giger. Deep learning in medical imaging and radiation therapy. *Medical physics*, 46(1):e1–e36, 2019.
113. Robert E Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2–3):135–168, 2000.

114. Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
115. Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 145–158. Springer, 2011.
116. Hoo-Chang Shin, Kirk Roberts, Le Lu, Dina Demner-Fushman, Jianhua Yao, and Ronald M Summers. Learning to read chest x-rays: Recurrent neural cascade model for automated image annotation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2497–2506, 2016.
117. Przemysław Skryjomski, Bartosz Krawczyk, and Alberto Cano. Speeding up k-nearest neighbors classifier for large-scale multi-label learning on GPUs. *Neurocomputing*, 354:10–19, 2019.
118. Newton Spolaôr, Maria Carolina Monard, Grigorios Tsoumakas, and Huei Diana Lee. A systematic review of multi-label feature selection and a new method based on label construction. *Neurocomputing*, 180:3–15, 2016.
119. Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, 2015.
120. Lei Tang, Suju Rajan, and Vijay K Narayanan. Large scale multi-label classification via metalabeler. In *Proceedings of the 18th international conference on World wide web*, pages 211–220. ACM, 2009.
121. Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pages 1345–1352, 2007.
122. Fadi A Thabtah, Peter Cowling, and Yonghong Peng. MMAC: A new multi-class, multi-label associative classification approach. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 217–224. IEEE, 2004.
123. Jimena Torres Tomás, Newton Spolaôr, Everton Alvares Cherman, and Maria Carolina Monard. A framework to generate synthetic multi-label datasets. *Electronic Notes in Theoretical Computer Science*, 302:155–176, 2014.
124. Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, volume 8, pages 325–330, 2008.
125. Grigorios Tsoumakas, Anastasios Dimou, Eleftherios Spyromitros, Vasileios Mezaris, Ioannis Kompatsiaris, and Ioannis Vlahavas. Correlation-based pruning of stacked binary relevance models for multi-label learning. In *Proceedings of the 1st international workshop on learning from multi-label data*, pages 101–116, 2009.
126. Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, volume 21, pages 53–59. sn, 2008.
127. Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, 2010.
128. Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.
129. Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *Machine Learning: ECML 2007*, pages 406–417, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
130. Naonori Ueda and Kazumi Saito. Parametric mixture models for multi-labeled text. In *Advances in neural information processing systems*, pages 737–744, 2003.

131. Shankar Vembu and Thomas Gärtner. Label ranking algorithms: A survey. In *Preference learning*, pages 45–64. Springer, 2010.
132. Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.
133. Willem Waegeman, Krzysztof Dembczyński, and Eyke Hüllermeier. Multi-target prediction: a unifying view on problems and methods. *Data Mining and Knowledge Discovery*, 33(2):293–324, 2019.
134. Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. CNN-RNN: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2016.
135. Xi Wang and Gita Sukthankar. Multi-label relational neighbor classification using social context features. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 464–472. ACM, 2013.
136. Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. ChestX-Ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017.
137. Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, and Ronald M Summers. TieNet: Text-image embedding network for common thorax disease classification and reporting in chest x-rays. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9049–9058, 2018.
138. Yunchao Wei, Wei Xia, Min Lin, Junshi Huang, Bingbing Ni, Jian Dong, Yao Zhao, and Shuicheng Yan. Hcp: A flexible CNN framework for multi-label image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1901–1907, 2015.
139. Havi Werbin-Ofir, Lihi Dery, and Erez Shmueli. Beyond majority: Label ranking ensembles based on voting rules. *Expert Systems with Applications*, 2019.
140. Jason Weston, Ameesh Makadia, and Hector Yee. Label partitioning for sublinear ranking. In *International Conference on Machine Learning*, pages 181–189, 2013.
141. Qingyao Wu, Mingkui Tan, Hengjie Song, Jian Chen, and Michael K Ng. Ml-forest: A multi-label tree ensemble method for multi-label classification. *IEEE transactions on knowledge and data engineering*, 28(10):2665–2680, 2016.
142. Xi-Zhu Wu and Zhi-Hua Zhou. A unified view of multi-label performance measures. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3780–3788. JMLR. org, 2017.
143. Yu-Ping Wu and Hsuan-Tien Lin. Progressive random k-labelsets for cost-sensitive multi-label classification. *Machine Learning*, 106(5):671–694, 2017.
144. Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4582–4591, 2017.
145. Jianhua Xu. Fast multi-label core vector machine. *Pattern Recognition*, 46(3):885–898, 2013.
146. Suping Xu, Xibei Yang, Hualong Yu, Dong-Jun Yu, Jingyu Yang, and Eric CC Tsang. Multi-label learning with label-specific feature reduction. *Knowledge-Based Systems*, 104:52–61, 2016.
147. Yan Xu, Liping Jiao, Siyu Wang, Junsheng Wei, Yubo Fan, Maode Lai, and Eric I-chao Chang. Multi-label classification for colon cancer using histopathological images. *Microscopy Research and Technique*, 76(12):1266–1277, 2013.
148. Fengqi Yan, Xin Huang, Yao Yao, Mingming Lu, and Maozhen Li. Combining LSTM and DenseNet for automatic annotation and classification of chest x-ray images. *IEEE Access*, 7:74181–74189, 2019.
149. Yan Yan, Ying Wang, Wen-Chao Gao, Bo-Wen Zhang, Chun Yang, and Xu-Cheng Yin. LSTM: Multi-label ranking for document classification. *Neural Processing Letters*, 47(1):117–138, 2018.

150. Z. Yan, W. Liu, S. Wen, and Y. Yang. Multi-label image classification by feature attention network. *IEEE Access*, 7:98005–98013, 2019.
151. Hao Yang, Joey Tianyi Zhou, Yu Zhang, Bin-Bin Gao, Jianxin Wu, and Jianfei Cai. Exploit bounding box annotations for multi-label object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–288, 2016.
152. Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. SGM: Sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3915–3926, 2018.
153. Yiming Yang and Siddharth Gopal. Multilabel classification with meta-level features in a learning-to-rank framework. *Machine Learning*, 88(1–2):47–68, 2012.
154. Yiyu Yao. The superiority of three-way decisions in probabilistic rough set models. *Information Sciences*, 181(6):1080–1096, 2011.
155. Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. Learning deep latent space for multi-label classification. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
156. Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning*, pages 3069–3077, 2016.
157. Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, and Jianfeng Lu. Multilabel image classification with regional latent semantic dependencies. *IEEE Transactions on Multimedia*, 20(10):2801–2813, 2018.
158. Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12(2):191–202, 2018.
159. Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.
160. Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.
161. Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.
162. Ping Zhang, Guixia Liu, and Wanfu Gao. Distinguishing two types of labels for multi-label feature selection. *Pattern Recognition*, 2019.
163. Wenjie Zhang, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha. Deep extreme multi-label learning. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 100–107. ACM, 2018.
164. Yuanjian Zhang, Duoqian Miao, Zhifei Zhang, Jianfeng Xu, and Sheng Luo. A three-way selective ensemble model for multi-label classification. *International Journal of Approximate Reasoning*, 103:394–413, 2018.
165. Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.
166. Yangming Zhou, Yangguang Liu, Jiangang Yang, Xiaoqi He, and Liangliang Liu. A taxonomy of label ranking algorithms. *Journal of Computers*, 9(3):557–565, 2014.
167. Yangming Zhou and Guoping Qiu. Random forest for label ranking. *Expert Systems with Applications*, 112:99–109, 2018.
168. Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320, 2012.
169. Feng Zhu, Hongsheng Li, Wanli Ouyang, Nenghai Yu, and Xiaogang Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5513–5522, 2017.
170. Yue Zhu, Kai Ming Ting, and Zhi-Hua Zhou. Multi-label learning with emerging new labels. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1901–1914, 2018.

Reinforcement Learning for Data Science

Jonatan Barkan, Michal Moran, and Goren Gordon

1 Introduction

In the realm of data science, where big data abound, most machine learning methods fall into one of the two categories: *supervised learning* in which labeled data exist, i.e., for each input in the dataset there is a known output; *unsupervised learning* in which no label exists and patterns in the data are sought after.

Reinforcement learning (RL) is a distinct category in the machine learning zoo, in which an agent *can act*, i.e., influence the environment it exists in, and receive rewards (scalar values). This field of machine learning is distinct from supervised learning, in that the true output is unknown, and from unsupervised learning, in that feedback is received from the environment, i.e., the reward.

Moreover, and more crucial, is that the learning process in RL involves *actions* and not just passive data streams. Thus, there are two major domains in which RL flourishes. The first domain is games: in a game, which can be a computer game [18] or not [22, 29], an agent has to make decisions regarding the actions that the game allows. Based on the state of the game and the action performed, the agent either receives rewards, e.g., points, stars, etc., or advances towards winning the game against an opponent [28]. The problem to be solved can thus be formulated as: given the current status of the game and the available actions, which actions should the agent perform that will maximize its rewards/chances of winning? Reinforcement learning algorithms have emerged as the dominant approach to solve these types of challenges, by playing millions of games in the computer and learning the optimal *policy*, i.e., which actions to choose to win the game.

J. Barkan · M. Moran · G. Gordon (✉)
Curiosity Lab, Department of Industrial Engineering, Tel-Aviv University, Tel Aviv, Israel
e-mail: goren@gorengordon.com

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_24

The second domain in which RL has flourished is robotics. In this field, the robot is the agent that, based on the input from its sensors, acts in the world and receives rewards based on the goals set out for it. In this way robots have learned to walk [4, 33], play ping-pong [21] and single-handedly solve a Rubik's cube [1]. Similar to the games domain, these amazing feats have been achieved by having the robot repeatedly try the task and actively exploring different policies, until convergence to the policy that achieves a satisfactory performance of the task.

Since conventional data science tasks involve a given dataset, with little to no option of actively and repeatedly making decisions, RL has not yet been a standard tool for data scientists. However, several interesting approaches have started to utilize the power of RL as part of the data science pipeline [19]. Moreover, with the emergence and proliferation of deep learning, deep reinforcement learning approaches to data science are bound to surface in the near future.

Hence, this chapter is aimed to introduce the reader to the basic principles, formulations, and algorithms of reinforcement learning (Sect. 2), give an example of its use in data science (Sect. 3), and explore the state-of-the-art approaches to deep reinforcement learning (Sect. 4). The chapter is not meant to give an exhaustive list of RL algorithms but rather gently present the principles and guidelines of how to approach a data science challenge, using RL tools.

2 Reinforcement Learning Formulation

In this section we first introduce the basic formulation of RL that will be used throughout the chapter. We then derive the famous Bellman equation, which is at the center of all RL algorithms. This is followed by an important dilemma of RL, namely the exploration-exploitation trade-off. The chapter ends with a brief account of the prominent classical RL algorithms, namely Q-learning, SARSA, and actor-critic approach.

We encourage the reader to first think of a concrete problem, in which you, the data scientist, or your algorithm, has the possibility to act, or change something in the world, and get a response. In the following section, we sprinkle the text with *[guiding questions]* to help you formulate your problem in RL terms.

2.1 Problem Formulation

Reinforcement learning is formulated as a Markov Decision Process (MDP), which is described by the following elements:

States: States describe the Markovian element of the formulation, i.e., the state of the system is a full account of all the required information about the system at a given time. All decisions and rewards will be based on the current state of the

system. States can be multi-dimensional, discrete, or continuous. States are denoted by $s_t \in S$, where t denotes time and S is the entire state space.

[What are the states of your problem? How can you define the status of your system in concrete and quantitative terms? Are your states continuous or discrete? What is the dimensionality of your state space?]

Actions: Actions describe the possible decisions that the agent can make at any given time. They are at the core of RL that separates it from other machine learning algorithms. In RL, an agent can act upon the world and possibly change its state. Actions can be multi-dimensional, discrete, or continuous. Actions are denoted by $a_t \in A$, where A is the entire action space.

[What are the actions? What can you or your algorithm do? Do the actions change depending on the state, or is the set of actions constant? Are your actions continuous or discrete? What is the dimensionality of your action space?]

Rewards: Rewards are scalar values that the agent receives from the environment. They are at the core of the problem the agent attempts to solve and represent utility or value for a given state and action. Higher reward values are better than lower rewards. The RL problem will be formulated as a maximization problem over the rewards (see below). Rewards are scalar and are denoted by $r_t \in \mathbf{R}$.

[What is the goal? How do you quantify when your problem has been solved? Can you quantify when the system progresses towards the goal, or only at the end?]

The world, or environment, the agent is embedded in is represented by two functions. The first is the *state-transition function* which represents how the state of the system changes over time and due to actions performed by the agent. This is given by:

$$P^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{1}$$

As can be seen, the state-transition function can be probabilistic, i.e., the state at the next time step is conditional on the current state and the action the agent performed. The state-transition function thus represents two possible contributions to a changing state, namely one is passive change, i.e., states that change regardless of any actions performed by the agent (for example, an apple falling from a tree) and the other is active, i.e., state's change induced by the actions of the agent (for example, a robot shaking the tree).

[Is there inherent dynamics in your system, even without you/your algorithm within it? Do you know how the system changes once you/your algorithm acts? Are the changes in the system's states deterministic or stochastic? Can you quantify the stochasticity in your system? If these transitions are unknown, there are specific algorithms that deal with it (see below).]

The second function that describes the world is the reward function. This function dictates the actual problem to be solved or the goal formulation for the agent. For each state and performed action, there is an expected reward:

$$R^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \tag{2}$$

[Is there a specific function to represent your goal? Does it depend on the states only, or does a specific action result in rewards? Is your reward deterministic or stochastic?]

Another integral part of the reward formulation is an important concept known as *discount*, denoted by $\gamma \leq 1$. The discount represents the notion that later rewards worth less than reward now. Thus, $\gamma = 1$ means that there is no discount, and all rewards, present and future, are the same, whereas $\gamma < 1$ means that reward in the next time step is worth γ-percent from the same reward now. An extreme example is $\gamma = 0$, which represents a *myopic* reward, i.e., only the current reward matters, as all future rewards are completely discounted and are not important for decision making.

[Is it important to solve the problem quickly, or just that it is solved, no matter how fast? Are there time-constraints to the solution?]

The agent itself is represented by the *policy*, which is a probabilistic function that maps states to actions. In other words, the agent's decision making process is governed by the policy, which, given the current state, assigns probabilities to each possible action. The policy is denoted by:

$$\pi(s, a) = Pr\{a_t = a | s_t = s\} \tag{3}$$

[Can you define a mapping from states to actions?]

Once all of the aforementioned definitions are done, we move to the formulations of the problem and then to ways of solving it. Implementing them is more straightforward, if the definitions of the problem have been done correctly.

We now can formulate the MDP problem. We define the value function as the expected accumulated discounted reward:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\} \tag{4}$$

The value function represents all the future accumulated discounted rewards, given we started at state s and followed policy π. In other words, it represents what are the rewards the agent is expected to receive if it started from a certain state and followed a specific policy.

The optimal value function is achieved by maximizing over all possible policies and is denoted by:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{5}$$

The policy that maximizes the value function represents the optimal solution to the MDP problem.

Another very important formulation is given by the *Q-function*. The Q-function is the expected future accumulated discounted rewards given the current state and *the current action performed* for a given policy:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t|s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\right\} \tag{6}$$

Similarly to the value function, one can define the optimal Q-function over all policies to be:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{7}$$

Given these formulations, the question is how to find the optimal policy.

2.2 Bellman Equation

Let us try to understand the "dynamics" of the MDP. We do this by reformulating the optimal value function:

$$V^*(s) = \max_a Q^{\pi}(s,a) \tag{8}$$

$$= \max_a E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\right\} \tag{9}$$

$$= \max_a E_{\pi}\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_t = s, a_t = a\right\} \tag{10}$$

$$= \max_a E_{\pi}\left\{r_{t+1} + \gamma V^*(s_{t+1})|s_t = s, a_t = a\right\} \tag{11}$$

$$= \max_a \sum_{s'} P^a_{ss'}\left(R^a_{ss'} + \gamma V^*(s')\right) \tag{12}$$

This amazing recursive equation, also known as the Bellman equation, shows that the optimal value function can be computed recursively. A similar equation exists for the Q-function:

$$Q^*(s,a) = \sum_{s'} P^a_{ss'}\left(R^a_{ss'} + \gamma\max_{a'} Q^*(s',a')\right) \tag{13}$$

$$Q^*(s_t,a_t) = r_{t+1} + \gamma\max_{a'} Q^*(s_{t+1},a') \tag{14}$$

From these equations, one can now calculate an important measure known as the *temporal difference error* or TD-error, denoted by δ. It is a measure of how far a given policy is from the optimal one:

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \tag{15}$$

For the optimal policy, given the Bellman equations, the TD-error will be equal to zero. The TD-error has a unique interpretation, namely it is the reward-prediction error. It measures how much the reward the agent received differed from the reward the agent expected to receive. If the agent has the optimal policy, it means that, due to the Bellman equation, it can predict the next reward. Thus, if the TD-error is not zero, the agent was surprised by the received reward, which in turn means that its Q-function is not optimal and does not represent that real Q-function. The TD-error can now be used to change and adapt the Q-function.

2.3 Q-Learning and SARSA

There are two algorithms that utilize the insights gained from the aforementioned formulations. The first is known as *Q-learning* and is a straightforward application of the TD-error. Given a Q-function, the algorithm slowly changes it given the TD-error:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \delta_t \tag{16}$$

where α is the learning rate. In other words, the agent's algorithm operates as follows:

1. Choose an action, a_t based on $Q(s_t, a)$ (see below)
2. Receive reward r_{t+1}
3. Calculate TD-error, δ_t, Eq. 15
4. Update the appropriate Q-function, Eq. 16

A very similar algorithm is called *SARSA*, the acronym of State-Action-Reward-State-Action. It was developed to overcome the problem of maximizing over all possible actions in Q-learning. In SARSA, the agent performs another action prior to updating the Q-function. Thus, the update is given by the actual new $Q(s_{t+1}, a_{t+1})$, i.e., given the selected action, and not optimizing over all possible actions:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right) \tag{17}$$

Thus, the SARSA algorithm for the agent operates as follows:

1. Choose an action, a_t based on $Q(s_t, a)$ (see below)
2. Receive reward r_{t+1}
3. Choose another action, a_{t+1} based on $Q(s_{t+1}, a)$ (see below)
4. Update the appropriate Q-function, Eq. 17

The two algorithms still have an undescribed component, namely how to choose an action based on the Q-function. This is addressed next.

2.4 Exploration-Exploitation

The agent is tasked to find the optimal policy that will grant it the most future accumulated discounted rewards. However, in order to find out what it is, it has to *explore* the environment, i.e., visit many states and select many actions. On the other hand, if the agent has already learned a proper Q-function, it should *exploit* the knowledge already gained to achieve maximal value.

This is the verbal formulation of the exploration-exploitation trade-off [30]. When should the agent explore, to find out new possible better policies, and when should it exploit the knowledge already gained and try to maximize the rewards accumulated?

There are two common policies that address this issue, namely ϵ-greedy and softmax. In the ϵ-greedy algorithm, the agent explores randomly (with uniform distribution) ϵ percent of the time and the rest exploits its knowledge, i.e., chooses the action that gives the maximal $Q(s, a)$, with ϵ having a typical value of 0.1. In other words, 10% of the time steps, the agent chooses randomly which action to perform, thus exploring new options in search of a better policy than it already has, and 90% of the time it exploits the policy it has already learned from the interaction with the environment, to try and accumulate as much reward as possible. Best practices dictate that ϵ should start with a large value and decrease with time, as confidence in the learned Q-function increases.

The second common policy is called softmax and it is formulated as:

$$\pi(s, a) = \frac{e^{\beta Q(s,a)}}{\sum_{a'} e^{\beta Q(s,a')}} \tag{18}$$

where $\beta = 1/T$ is called the inverse temperature (T) and controls the exploration-exploitation trade-off. This policy assigns probabilities based on the actual Q-function values: actions that are predicted to give higher values have higher probabilities. β, the inverse temperature, dictates how much exploration is performed: when $\beta = 0$, the temperature is infinite and the agent explores, with all actions having the same probability, regardless to their Q-value, i.e., uniform distribution over the actions; when $\beta \to \infty$, the temperature is zero and the agent exploits, with only the action with the maximal $Q(s, a)$ having probability one and the others probability zero. As with the ϵ-greedy algorithm, best practices indicate that β should start low and increase, to transform from high exploration to high exploitation.

2.5 On- and Off-Policy

Another important distinction between different RL algorithm is whether they are on- or off-policy. An on-policy algorithm means the algorithm requires the

application of the policy learned, i.e., it requires that the action be selected by the agent. An example of an on-policy algorithm is SARSA, as the update algorithm requires another action by the agent for the update of the Q-function.

An off-policy algorithm does not require to perform the action by the agent itself, but rather it can be supplied a sequence of state-action-rewards and update the policy by these. An example of an off-policy algorithm is Q-learning, as in the algorithm itself, only the current state, action, and reward are used, and the agent, for the update itself, does not need to perform another action.

2.6 Actor-Critic

Actor-critic (AC) algorithms are based on the simultaneous online estimation of both the policy, $\pi(s, a)$ (actor) and the value function (critic). Thus, the critic addresses a problem of prediction, whereas the actor is concerned with control. These problems are separable but are solved simultaneously to find an optimal policy, as in the aforementioned policy iteration.

However, in actor-critic algorithms, the policy and value function are *approximated* using function approximation, i.e., they are parameterized. Thus, for example, the value function and policy can be described by:

$$\hat{V}_t(s) = \mathbf{v}_t^T \mathbf{f}(s) \tag{19}$$

$$\hat{\pi}_t(s, a) = \boldsymbol{\theta}_t^T \boldsymbol{\psi}(s, a) \tag{20}$$

where $\mathbf{v}$ is a vector of the adaptable parameters of the critic and $\mathbf{f}(s)$ is a vector of its basis functions; $\boldsymbol{\theta}$ is a vector of the adaptable parameters of the actor and $\boldsymbol{\psi}(s, a)$ is a vector of its basis functions.

The updates to both are preformed simultaneously, based on the TD-error:

$$\delta_t = r_{t+1} + \gamma \mathbf{v}_t^T \mathbf{f}(s_{t+1}) - \mathbf{v}_t^T \mathbf{f}(s_t) \tag{21}$$

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \alpha_t \delta_t \mathbf{f}(s_t) \tag{22}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta_t \delta_t \boldsymbol{\psi}(s_t, a_t) \tag{23}$$

where α_t, β_t are time-dependent learning rates of the critic and actor, respectively. In these types of algorithms, the learning rate is also usually decreasing, where the learning rate of the actor should be lower than that of the critic, since the prediction problem should converge first, then dictating the convergence of the policy.

While there are numerous extensions to these algorithms, we next turn to a practical implementation for data science problems and then elaborate on the most prominent algorithms of deep reinforcement learning.

3 Curious Feature Selection

In the current section, we present an implementation of reinforcement learning to the subfield of Feature Selection. We first introduce it and then elaborate on the RL-related formulations that follow [19].

3.1 Feature Selection

In the machine learning subfield, feature selection is the process of selecting a subset of relevant variables to be used in model construction. Feature selection techniques are primarily intended to improve model prediction performances and runtimes, to reduce model overfitting, and to increase generalization [12]. Several feature selection algorithms exist, most of which use search techniques along with an evaluation index to find an appropriate feature subset. The most basic technique searches all possible subsets and selects the subset that minimizes the model error. However, this exhaustive search is not computationally effective, particularly in situations where a large number of features exist [6].

In the following example [19], we applied the concepts of intrinsically motivated autonomous agents to data structures. We implement the curiosity loop architecture for the task of learning the data structure and performing curious feature selection (CFS). First, the training data is divided into small episodes. In each episode, a new feature (action) is selected in an inner-loop based on the already selected previous feature (state), and the internal reward consists of a reduction in the learning model error [25]. The model runs in a loop on multiple episodes until it reaches convergence. The loop attempts to find a feature selection policy that optimizes the model accuracy. The entire process is completely autonomous in the sense that all actions and rewards are determined autonomously. The resulting policy is used to construct a new learning model on the entire training set and on an unseen testing set. We show that this architecture improves model accuracy compared to running the learning model on the entire dataset or using common feature selection algorithms.

3.2 Intrinsically Motivated Learning and the Curiosity Loop

The basic intrinsically motivated learning [25], also known as artificial curiosity [27] or the curiosity loop [9, 10, 11], is based on the reinforcement learning paradigm, which is composed of a learner (predictor) and an intrinsic reward. In each loop, the agent selects an action and obtains an internal reward, which is based on prediction errors. The agent's goal is to select an optimal action that maximizes its learning and to learn an action selection policy that maps states to actions. Each loop's

convergent dynamics leads to a specific behavior tightly related to the objective learnable correlation [9, 10, 11].

Because the goal of the basic curiosity loop is to autonomously and actively learn a correlation in the best way possible, the core of each loop is the learner, which attempts to map presented input–output pairs via an internally supervised learning algorithm. For each presented example, the learner acquires a prediction error (i.e., the difference between the expected output and the correct output) and the reward value is set to this error [25, 9, 10, 11]. Thus, the agent in the curiosity loop actively learns via the reinforcement of the prediction errors. The agent determines which new example is presented to the learner by selecting an appropriate action. Then, the learner produces the prediction error, which determines the intrinsic reward. The reward is translated into a value function used to update the agent policy, which is then used in the next loop to determine the next appropriate action to select [9, 10, 11].

3.3 States and Actions

As described in the previous section, the first important issue to resolve is the definition of the state- and action spaces. To recap, the basic reinforcement learning model consists of a set of environment and agent states $s \in S$ and a set of actions $a \in A$ that the agent can select.

In the CFS algorithm, both the state space and the action space are equal to the data feature space, that is, each state represents the previously selected feature and each action is the next feature to select. The state space contains one additional initial state s_0 that represents the state before any feature is selected.

$$s \in S : \{s_0, feature_1, feature_2, \cdots feature_n\} \tag{24}$$

$$a \in A : \{feature_1, feature_2, \cdots feature_n\} \tag{25}$$

Each state has a set of available actions $A_{available} \in A$ that represents the features not yet selected in the current loop.

3.4 Learner and Internal Reward

Once the states and actions are defined, the next important definition is the reward. In CFS, we use intrinsic reward, in other words, reward that depends on another element within the algorithm and not supplied by the user. The intrinsic reward will be determined by improvement of the learned model, i.e., the learner.

The learner can be any supervised machine learning model. The model attempts to learn an input (i)-output (o) transformation: $L\,(O \mid I : \chi)$ where χ represents the

parameters of the machine learning model. The learner finds the parameters χ that best minimize the generalization error:

$$L = \arg\max_{\chi} \left(\sum_{i} \left(O_{model}\,(i) - O_{real}\,(i) \right)^2 \right) \tag{26}$$

At each time t, the learner is executed on the episode data, using the selected feature (action) and all the other features that were selected in the current episode. After each execution, the model output O_{model} is compared to the real output O_{real}, resulting in a prediction error e_t. The prediction error e_t is compared to the previous error e_{t-1}, and the reward r_t is defined as the change in the error: larger changes in the error result in larger rewards.

$$r_t = e_{t-1} - e_t \tag{27}$$

To calculate the reward for the first selected feature, we define an initial error, which is a prior error assumption. The change in the error represents the importance of the last selected feature and the information it adds to the model. Thus, when the agent selects a feature that adds valuable information to the model and, thus, reduces the total error, the reward is high.

In the CFS algorithm, we do not know the transition function. However, this is not an issue as we use algorithms that do not assume this knowledge. Once all the definitions are done, we move to the description of the value function and policy.

3.5 State-Action Value Function and Policy

The received internal reward is used to update a state-action value function that yields the expected utility of adding a specific feature (action) to a given feature (state).

The policy is the rule to follow when adding a specific feature (action) to a given feature (state). Starting with the initial state, the optimal policy can be constructed by simply selecting the feature (action) with the highest value above some threshold, setting this feature as the next state and repeating this operation until no more actions are available.

$$s_{t+1} = \arg\max_{a}(Q(s_t, a_t)) \tag{28}$$

3.6 Exploration vs. Exploitation

To balance between leveraging the model's knowledge in each state by selecting the action with the highest estimate action-value function (exploitation) and exploring the uncharted features space, the epsilon-greedy algorithm is used, and an epsilon parameter ϵ is defined [16]. The action with the highest estimate action-value function is selected with a probability of $1 - \epsilon$, while a random action is selected with a probability of ϵ.

To improve exploration during the first few episodes, the parameters ϵ and α (learning rate) are set to high values and then automatically reduced based on how many iterations have been completed [7]. In addition, an optimistic Q-learning [8] approach is used, where the Q-values are initialized to 1. This creates a greedy policy with respect to the Q-values for the first few episodes. The Q-value is updated with the actual value of the reward the first time each state-action is selected.

3.7 Experimental Setting

We executed the CFS algorithm on the diabetes datasets.

3.7.1 Diabetes Dataset

The Diabetes data was extracted from the Health Facts database (Cerner Corporation, Kansas City, MO), a national data warehouse that collects comprehensive clinical records from hospitals throughout the United States. It was submitted to the UCI Machine Learning Repository on behalf of the Center for Clinical and Translational Research, Virginia Commonwealth University [31]. After data preprocessing, the dataset contained 100,000 instances with 44 features describing encounters with diabetic patients, including their demographics, diagnoses, diabetic medications and number of visits in the year preceding the encounter.

3.7.2 Data Preparation

As a preliminary step, it is split into a training dataset (90%) and testing dataset (10%). The training dataset is used to train the CFS algorithm and select the policy for the feature subset. The testing dataset is used to rebuild the model based on the subset of selected features and to evaluate the model on new data. On each dataset, the CFS algorithm was tested with 2 learners (Decision Tree and Naive Bayes). Specifically, these learners were selected since they are applicable for both categorical and numerical features types. The discount factor was set to 0. Each experiment was repeated 30 times and the average result was taken.

3.7.3 Comparison Algorithms

We compared the CFS algorithm results on the testing dataset with 4 traditional, popular and commonly used feature selection algorithms. The algorithms represent each one of the feature selection categories: including the Sequential Forward Selection (SFS) wrapper method [17], 2 filter methods: the Variance Threshold method, which removes all features whose variance does not meet some threshold and the Chi-Square test [15], and one embedded method based on the GINI importance score for decision tree [5] that measures the average gain of purity by splitting a given variable.

3.7.4 Results

On the Diabetic dataset, the traditional feature selection methods reduce the number of features from 44 to 13–14 features (Sequential Forward Selection),20 features (Variance), 23 features (DT GINI importance), 26 features (Chi-square test—60%), and 35 features (Chi-square test—80%), and in some cases improved the model accuracy compared to the baseline. However, in other cases it did not change the accuracy or reduce the accuracy. For the decision tree model, the baseline accuracy was 46.08%. The best traditional method (Sequential Forward Selection) achieved an accuracy of 57.17% and improved the baseline accuracy by 11.09%. The CSF algorithm's best and most frequent policy (22 out of 30) included only one feature and achieved an accuracy 11.27% higher than the baseline. For the Naive Bayes model, Sequential Forward Selection methods achieved accuracy of 58.12% however, the other traditional feature selection methods did not change the model accuracy (45.06%). The CSF algorithm's best and most frequent policy (5 out of 30) which included only 3 features, achieved an accuracy of 58.17%, and improved on the baseline method by 13.11%.

3.8 Summary

The CFS algorithm with the RL approach is more efficient, run-time-wise, when adding more features or increasing the data size [19]. Moreover, it suffers less from over fitting and provides insights on the data, due to multiple policies, in the form of importance of features and their combinations.

These impressive results were obtained using classical RL algorithms. We next dive into deep reinforcement learning, an extremely successful extension of function approximation of the different components of RL, e.g., policy, value function, and Q-function.

4 Deep Reinforcement Learning

In this section we introduce the concept of deep reinforcement learning (DeepRL) in a formal way and then present important issues that arise specifically with data science related topics. This section is not meant to be an exhaustive review of DeepRL algorithms, but rather a list of guidelines on important decision criteria on how to choose the appropriate algorithm.

We first give a brief introduction to deep neural networks and their direct application to RL and then focus on challenges of DeepRL and some of their solutions.

4.1 Introduction to DeepRL

Deep Reinforcement Learning is a variant of RL which utilizes Artificial Neural Network (ANN) as the function approximator. ANNs, recently re-popularized under the paradigm of machine learning through the use of the backpropagation (backprop) algorithm, is the name given to the class of computational models that are represented as weighted directed graphs, where each node is equipped with a real-valued function.

One can group the nodes into the following groups: (1) Input Nodes—nodes that have no edges going into them. (2) Hidden nodes—nodes that have edges going into them and out of them. (3) Output Nodes—nodes that have no edges going out of them. The graph itself, without the weights or edges connecting the nodes, is called the *Architecture* of the network.

Each ANN is a model of a function, where the input is inserted to the Input Nodes and the output retrieved from the Output Nodes. Each node performs the computation in 2 steps:

1. Summing all the outputs of the nodes going into it, weighted by their respective weights
2. Applying its associated function, called activation function, on the result of the last step

The computation of the network means performing this process on all the nodes in the graph.

If the Architecture is not acyclic, a problem with this formulation may occur as there will be multiple possible sequences for performing the computation, and they might result in different outputs. For this reason, most of the current Architectures are acyclic. As a result, we can think of the architecture in an hierarchical way, with each layer being comprised of nodes receiving inputs from the previous layer, with the first layer the input layer, the last layer the output layer, and all the layers in between called Hidden layers.

The ANNs weights are learned using back-propagation to perform Gradient descent. Backprop is an algorithm for traversing the error of the network and assigning "blame" (the gradient value) to every edge in the network so it can "learn" (by adjusting its corresponding weight). When an ANN has several hidden layers (initially more than 3, but today at least 5) it is called a Deep Neural Network (DNN).

Even though there is currently no accepted mathematical or computational explanation for their success, empirical results show that DNNs are very good at solving several tasks that, currently, other method fail to solve. The most successful applications are in the fields of computer vision (e.g., face detection, face recognition, face generation, (multi) object detection, image segmentation), and natural language processing (e.g., speech recognition, translation).

More formally, the goal of a neural network is to approximate function f. For a neural network F with set of weights W we annotate the corresponding function as F_w. We can introduce networks into the learning paradigm by thinking of the weights as parameters, $w \rightarrow \theta$. Now our network is F_θ and the problem of function approximation becomes the problem of finding a set of parameters Θ such that F_θ best approximates our desired function f.

We use the DNN to approximate and learn the Q-function (or actor-critic functions), by using the TD-error as the error signal for the network.

4.2 Difficulties Relevant to DeepRL

4.2.1 Sample Efficiency

While successful, deep learning has its limitations, first and foremost is the amount of data it needs. For a deep network to be able to generalize well it must have enormous data resources. Deep learning leverages the statistics of the data it consumes, thus it must encounter numerous variations in its training data to be able to work and generalize to the real world. For a few tasks, e.g., face/object detection, there are huge datasets available online. However, for most other tasks, data, especially free and available data, is hard to come by.

This is doubly true for RL tasks. At the extreme, robots are real-world entities and as such perform actions in the real world, thus making the creation of a large set of examples (e.g., for SARSA) a tedious and very time consuming task. This can be partly solved by simulation, but simulations are not the real world and models trained on them might not transfer properly to a real environment.

Even if the RL agent is within a computerized environment, it can only perform a single action at a time. As a result, a single agent can only "see" one possible trajectory. One option is to use multiple agents within the same environment, but that only partially solves the problem: the size of the trajectory space, i.e., all possible sequences of state-action pairs. This space size is exponential with respect to the action and space sizes. Moreover, the time horizon plays a role too, as some

algorithms use time horizon greater than 1. To summarize, designers of algorithms in DeepRL must put sample efficiency at their front and center.

4.2.2 Sparse Rewards

DeepRL, as any RL, requires rewards in order to adapt and learn the policy. However, in many scenarios the reward is extremely sparse temporally, i.e., rewards are received very infrequently. For example, in some computer games, the agent does not get points during each level of the game, but only a "win/lose" for each level. This is very sparse, since the agent has to perform several actions in order to complete the level, for which it does not get any reward.

Similarly, in robotic implementations, the scenario is usually defined as a "success/failure" for each attempt and not a continuous reward for each decision made by the robot. This issue continues to plague the DeepRL field, with some attempts to overcome it, using curiosity, or intrinsic rewards [3].

4.2.3 Correlation between Data Points

Other, non-trivial problems exist too, but the field has been making inroads in addressing them. One such problem is that of correlated samples (also contributing to the sample efficiency problem). As of today, all of DNNs success leverages the Stochastic Gradient Descent (SGD) optimization algorithm and its variants. This approach to GD approximates the gradient by sampling a random sample of the training data at each iteration. For the stochastic sample to be a close (unbiased) estimation of the real gradient of the network at that time step the samples need to be as uncorrelated as possible. However, in RL the agent gathers samples directly from the environment and in small time intervals, resulting in highly correlated observations. For example, two consecutive states will most surely be very "close" to each other (under any reasonable metric of the state space). Thus, if we want it to learn on the fly (online), the samples will almost certainly be highly correlated.

4.2.4 Ground Truth Unavailability

Another difficulty is with respect to ground truth. DNNs success has come in the paradigm of Supervised Learning, where gradients are calculated with respect to an error value. That error is calculated by applying some metric on the output space. Specifically, it needs the ground truth, i.e., the correct output the model should have predicted. The ground truth is also referred to as target. Alas, in RL that ground truth is usually unavailable or intractable to calculate. For value functions it is the expected sum of discounted reward (intractable) and for policy methods that is an optimal policy (unavailable).

These obstacles should be added to all the previous difficulties that effect regular RL, e.g., exploration-exploitation.

4.2.5 Benchmarks

Benchmarks are a touchy subject for every task and especially so for DeepRL. How can we define an environment that when testing against it will tell us how generalizable our model is? Usually in Machine Learning (ML) the method is to split the available data to 2 parts, train and test. On the training chunk we train new models and compare/validate them and then judge the model's generalization capabilities against the test chunk. However, because of sample inefficiency and the mastering effect (due to the exploitation process), what should DeepRL agents generalize? As a result, DeepRL is the only field that tests itself against environments that are highly similar to the training environments.

For example, the most prevalent framework in which DeepRL research is done is a set of ATARI games. This is a collection that has very specific resolution, very limited set of actions (due to them being designed to be played with the same controller) and the objective is to move up in the levels. As a result, the main way of splitting the environments is by levels. But different levels are highly correlated with respect to color distribution, world dynamics, direction of game development (in some games the character always needs to move to the right to finish), and much more.

4.3 DeepRL Algorithms

DeepRL is a nascent field that is advancing rapidly. Several basic algorithms now follow, in order to give the reader an entry point to the field and the way some of the aforementioned challenges are resolved.

4.3.1 Deep Q-Networks (DQN)

The first to emerge as a good framework was [18]'s Deep Q-Network, which we will dive into in order to get an example of how these difficulties can be resolved.

Deep Q-Networks (DQN), as the name suggests, are based on the Q-learning paradigm/algorithm, found earlier in the chapter. Basically, it replaces the state-action table that is used in Q-learning with a Deep Neural Network.

First, we must set the scene of the problem, i.e., describe the environment, the state and action spaces and the reward.

DQN was initially built to play ATARI games, e.g., Breakout, were it plays as the paddle. As a result, the action space is small and is limited by the joystick used. For example, the paddle can move left, right, or stay put at each time step.

While the goal of the game is to pass each level, it makes for a bad reward function. If we choose the reward to be passing a level then we will get a very sparse reward, as it will come only at the end of each level, either win or loss, and as you may recall, a sparse reward function is very hard to "solve" with RL. Thankfully, there is a score attached to each block removed/hit. That score can be very helpful because: (1) A higher score correlates well with passing the level, and (2) It is much less sparse, as there are many blocks in each level. This overcomes the reward function conundrum because we get a non-sparse reward that can help us win levels.

As for the state space, we want to take advantage of the DNN paradigm. The most successful algorithm uses images where, specifically for the first layers, we use convolutional layers, allowing us to define the state space as raw pixels. But due to the actions being the size of 3, they are massively "smaller" than a state. That is circumvented by a smart architecture, where the input is the current state (actually 4 successive frames, not just 1) and the output is a probability distribution on the actions (using softmax to get the distribution). At any state, the agent "chooses" an action by drawing from the output's distribution on the actions.

That explains how the network behaves like an agent in the game, but we need to address one last thing, which is to explain how the network learns a good policy, that is, what is the network's optimization/minimization objective.

4.3.2 Q-Networks and Target Networks

For this we recall the Q-learning algorithm. In it, we used the Bellman equation to extract the Temporal Difference error. Stochastic Gradient Descent uses batches instead of the entire data, so the "natural" objective would be to minimize that, i.e., the target value is defined as $r_{t+1} + \gamma \cdot \max_{a \in A} Q(s_{t+1}, a)$.

Notice that the target uses Q, hence we need to be careful not to use the gradient with respect to it. But using the same network to compute both values creates an issue. Due to the max operator in the TD-error (Eq. 15), and the overestimation of the value function that learning from a moving target can create, the network might get "stuck" with a bad policy.

Because of this, in DQN they used two Q-networks, one as the true network and the other for computing the target value (which is called, unsurprisingly, a Target network).

Several target network versions are used in DeepRL, and they go under the name Double DQN. In the original paper [13], the method was to have two Q-networks and to use them in the role of learning and target interchangeably, i.e., every "couple" of iterations they switched between the roles. Another approach is to have only one network, but to keep an old copy of it to be used as the target [32]. In this method, they updated the "old" copy every "couple" of episodes in order for it to not go "stale." Another approach is, resembling the first, to use two networks, but unlike the original, both networks perform learning at the same time, and the target value estimation takes the minimum value between the two networks.

4.3.3 Experience Replay

To help solve the problem of high correlation between samples, [18] introduced the Experience Replay (ER) Buffer. Inspired by the brain, the buffer contains a large collection of past experiences. This makes the algorithm off-policy, as during the game itself the agent does not learn but rather plays according to its current policy. In DQN, each sample in the buffer is comprised from, at least, the tuple $(s_t, a_t, r_{t+1}, s_{t+1}, e)$, where e is a Boolean value that tells us if the sample was taken right before the episode terminated (because then the estimation of the Q-value is just the reward). Usually a memory buffer holds 50k samples, and some may be as big as several millions.

Experience replay has several versions. In [18] the researchers used a uniform distribution to sample the memory. In priority experience replay [26], they sampled the memory with a distribution extracted according to the last TD-error (it is updated every time it is sampled) that corresponds to the sample.

An important note: One should not be tempted to draw samples in a non-random way. Remember that the learning algorithm is a stochastic variant of Gradient Descent and therefore our sampling technique must be stochastic, i.e., random sampling. Otherwise the algorithm might "forget" the low error data points it learned—the ones it was correct on—as they would not be sampled almost at all.

Ref. [2] leveraged the fact that the learning happens offline to use hindsight in setting the goal. Essentially, the goal, and the associated reward r_g (see [2] for their goal-to-reward formulation), can be chosen retroactively, and one can assume that the state arrived at is exactly the state the agent wanted. In practice, because the goal affects the policy, sample a goal for the agent to pursue in the next episode and proceed to "playing" the episode and storing $((s_t||g), a_t, r_g, (s_{t+1}||g))$ in the buffer ($||$ is concatenation of the state and the goal, possible because both are elements of the state space). When the episode is over, start populating the replay buffer by sampling other goals and adding the corresponding tuple as if that was the true goal, i.e., change the reward and the concatenated states $(r_g \rightarrow r_{g'}, s||g \rightarrow s||g')$.

Lastly, it would be nice to be able to learn from good agents. Those can be either human agents or, if unavailable, big lumbering agents (the teacher-student construct in DNN, where we first get a very deep architecture to learn to play and then, in order to downsize the amount of memory used to store the network, we deploy a smaller network that learns from the bigger network). This is called demonstration/imitation replay buffer [14]. In it, in addition to the agent's experience, we populate the buffer with examples collected from the good agent.

We have presented several state-of-the-art solutions to the challenges of DeepRL. There are numerous extensions to the algorithms we have presented, which are beyond the scope of this chapter. Furthermore, the field is developing rapidly, with dozens of new algorithms every month. We refer the interested reader to the following websites [23, 24] and reviews on deep reinforcement learning algorithms [20].

5 Summary

In this chapter we have introduced the basic formulation of reinforcement learning. While there are only a few direct uses of RL in the field of data science, we have detailed one such use in the case of Curious Feature Selection. The proliferation of deep learning and its direct extension, DeepRL, promises to supply novel algorithms which are directly applicable to data science.

One way to think of how to implement DeepRL in data science pipelines is to consider the data scientist as the agent and model her decision making. Thus, for example, in Feature Selection, previously the data scientist had to select which features were important or not. In CFS we have used RL, with rewards proportional to better models, to efficiently learn which features to select. One may attempt to apply this approach to other decisions in the data scientist profession.

References

[1] F. Agostinelli, S. McAleer, A. Shmakov, and P. Baldi. Solving the Rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, 1(8):356–363, 2019.

[2] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in neural information processing systems*, pages 5048–5058, 2017.

[3] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[4] H. Benbrahim and J. A. Franklin. Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, 22(3):283–302, Dec. 1997.

[5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, Jan. 2014.

[7] C. Darken, J. Chang, and J. Moody. Learning rate schedules for faster stochastic gradient search. In *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*, pages 3–12, Aug. 1992.

[8] E. Even-Dar and Y. Mansour. Convergence of optimistic and incremental Q-learning. In *Advances in neural information processing systems*, pages 1499–1506, 2002.

[9] G. Gordon and E. Ahissar. Hierarchical curiosity loops and active sensing. *Neural Networks*, 32(Supplement C):119–129, Aug. 2012.

[10] G. Gordon, E. Fonio, and E. Ahissar. Emergent Exploration via Novelty Management. *Journal of Neuroscience*, 34(38):12646–12661, Sept. 2014.

[11] G. Gordon, E. Fonio, and E. Ahissar. Learning and control of exploration primitives. *Journal of Computational Neuroscience*, 37(2):259–280, Oct. 2014.

[12] I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.*, 3:1157–1182, Mar. 2003.

[13] H. V. Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[14] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, and I. Osband. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[15] I. F. Imam, R. S. Michalski, and L. Kerschberg. Discovering attribute dependence in databases by integrating symbolic learning and statistical analysis techniques. In *Proceeding of the AAAI-93 Workshop on Knowledge Discovery in Databases, Washington DC*, 1993.
[16] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *J. Artif. Int. Res.*, 4(1):237–285, May 1996.
[17] J. Kittler. Feature set search algorithms. *Pattern recognition and signal processing*, 1978.
[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]*, Dec. 2013. arXiv: 1312.5602.
[19] M. Moran and G. Gordon. Curious Feature Selection. *Information Sciences*, Feb. 2019.
[20] S. S. Mousavi, M. Schukat, and E. Howley. Deep Reinforcement Learning: An Overview. In Y. Bi, S. Kapoor, and R. Bhatia, editors, *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, Lecture Notes in Networks and Systems, pages 426–440, Cham, 2018. Springer International Publishing.
[21] K. Muelling, A. Boularias, B. Mohler, B. Schölkopf, and J. Peters. Learning strategies in table tennis using inverse reinforcement learning. *Biological cybernetics*, 108(5):603–619, 2014.
[22] H. C. Neto, R. M. S. Julia, G. S. Caexeta, and A. R. A. Barcelos. LS-VisionDraughts: improving the performance of an agent for checkers by integrating computational intelligence, reinforcement learning and a powerful search method. *Applied Intelligence*, 41(2):525–550, Sept. 2014.
[23] OpenAI. Key Papers in Deep RL.
[24] OpenAI. Kinds of RL Algorithms.
[25] P. Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286, Apr. 2007.
[26] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized Experience Replay. *arXiv:1511.05952 [cs]*, Nov. 2015. arXiv: 1511.05952.
[27] J. Schmidhuber. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.
[28] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, and T. Graepel. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.
[29] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. v. d. Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, Oct. 2017.
[30] A. Simpkins, R. De Callafon, and E. Todorov. Optimal trade-off between exploration and exploitation. In *2008 American Control Conference*, pages 33–38. IEEE, 2008.
[31] B. Strack, J. P. DeShazo, C. Gennings, J. L. Olmo, S. Ventura, K. J. Cios, and J. N. Clore. *Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records*. 2014.
[32] H. Van Hasselt, A. Guez, and D. Silver. Deep Reinforcement Learning with Double Q-Learning. In *AAAI*, volume 16, pages 2094–2100, 2016.
[33] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne. Feedback Control For Cassie With Deep Reinforcement Learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246, Oct. 2018. ISSN: 2153–0858.

Adversarial Machine Learning

Ziv Katzir and Yuval Elovici

1 Introduction

Machine learning techniques are used to induce models for various tasks. Until recently, the prevailing assumption was that the users of these models would not attempt to deliberately mislead them. However, a relatively new and extremely active research domain known as Adversarial Machine Learning studies machine learning algorithms in the presence of an adversary. Its main focus is understanding the susceptibility of machine learning algorithms to specially crafted inputs, referred to as *adversarial examples* or *adversarial perturbations*. These inputs are designed to mislead the learning algorithms so they will come to a wrong conclusion, while being indistinguishable from valid input. Initially, the term "indistinguishable" has referred to the human perception; however, in later works it was extended to include computerized detection methods as well (such as in the case of identifying adversarial perturbations against a cyber malware detection software).

Adversaries can either try to influence the learning process itself by *poisoning* the training dataset or alternatively aim to mislead an already trained model. This chapter is mainly focused on attacks against fully trained models, which are sometimes referred to as *evasion* attacks. Adversarial machine learning research includes methods for creating adversarial perturbations, methods for defending against adversarial perturbations, and research aimed at explaining the susceptibility of machine learning models to these perturbation patterns.

Early research on adversarial machine learning was mainly theoretical and inapplicable to real world systems, hence the entire domain was considered esoteric or even anecdotal. However, a series of discoveries in 2013–2016 quickly changed

Z. Katzir (✉) · Y. Elovici
Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel
e-mail: zivka@post.bgu.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_25

that perception and paved the way for the implementation of practical attacks against real world systems. Today, model susceptibility to adversarial examples is one of the main issues concerning the AI community.

Szegedy et al. (2013) showed, for the first time, that deep neural networks are susceptible to adversarial perturbations, much like traditional machine learning algorithms. Not long after, Goodfellow et al. (2014) discovered a simple, computationally efficient method for creating adversarial examples. As part of that work, the researchers also proposed *adversarial retraining*, the first defense mechanism against adversarial examples, and in doing so, marked the beginning of an arms race of attacks and defenses. Two years later Kurakin et al. (2016b) demonstrated how adversarial examples could be used in the physical world. Together, these three studies created a sense of urgency within the research community, and as a result, the amount of research attention dedicated to this field has dramatically increased.

In this chapter, we follow the evolution of the adversarial machine learning domain through the lens of the literature. We start with the early methods of attack and defense and conclude with recent discoveries and outstanding research questions. We review notable studies and discuss their contribution to the understanding of this phenomenon. Most of the works surveyed in this chapter deal with neural networks and, more specifically, neural network-based classifiers, but we do not want to imply that adversarial machine learning is limited to these types of learning models. On the contrary, adversarial machine learning has been shown to affect a wide variety of model types and learning tasks. Our choice of surveyed papers simply reflects the fact that, the vast majority of research performed in this domain has used neural network-based classifiers to demonstrate attack and defense methods. Still, most of the principles and methods discussed in this chapter are also applicable, with minor adaptations, to other machine learning tasks. The attacks and defense mechanisms mentioned here can easily be applied to other types of neural networks, or even to different machine learning algorithms.

2 The Very Early Works

The term *adversarial machine learning* was first coined in (Huang et al., 2011) to denote "the study of effective machine learning techniques against an adversarial opponent." The same work also introduced a taxonomy of attack methods, fostering an in depth discussion of the associated risks and mitigation methods. When a machine learning algorithm faces an adversary, one can no longer rely on the *data stationarity assumption*. This assumption, which serves as the basis of all machine learning algorithms, assumes that the statistical properties of the data used to train a model are identical to those of the data observed during inference. The existence of an opponent is therefore what distinguishes adversarial machine learning as an independent research domain.

Initial interest in the domain of adversarial machine learning was sparked when machine learning algorithms were put to use in commercial fraud prevention and

cyber defense applications. In those cases, the existence of an adversarial opponent is a fact of life, as opposed to a theoretical concept. As a result, many early research studies used spam filtering, which was among the first cyber defense applications addressed by machine learning, as their test case (Kołcz and Teo, 2009; Dekel et al., 2010).

Early research (2011–2014) mainly focused on traditional machine learning algorithms, such as the support vector machine (SVM) classifier, and specific clustering algorithms (Biggio, Nelson et al., 2012; Biggio, Corona et al., 2013; Biggio, Pillai et al., 2013). Game theory related research also had a major influence on the adversarial machine learning domain in that period (Brückner et al., 2012; Wang et al., 2014). Although both SVMs and game theory did not play a major role in later adversarial machine learning research, two main concepts that emerged from those early works continue to influence adversarial machine learning research to this day:

1. The use of loss functions for defining the utility of either the attacker or the defender
2. The use of standard "solvers" or optimizers for crafting adversarial attack patterns

By 2014, machine learning research in general began to focus more heavily on deep neural network-based algorithms, and such algorithms continue to dominate machine learning research today. This triggered a similar pattern in the domain of adversarial machine learning, and as a result, the vast majority of studies surveyed in this chapter use deep neural network-based classifiers as their use case. Still, most of the principles and methods we survey are applicable in a wide range of use cases and machine learning algorithms.

3 The Evolution of White-Box Attack & Defense Methods against Deep Neural Networks

White-box adversarial attack methods against neural networks were the first adversarial attack methods to be discovered, and they account for the majority of all attack methods published to date. These methods assume that the attacker has complete knowledge of the attacked network's architecture and parameters. This all-powerful attacker model greatly simplifies the process of crafting adversarial examples. In fact, the basic white-box attack concept strongly resembles the process of training a neural network.

We start our journey into the evolution of adversarial attacks by studying early white-box methods. As will be seen later in this chapter, the principles associated with these methods serve as the foundation for many of the more advanced attack methods, including those that assume a more restricted attacker model. We then move on to describe the arms race triggered by those methods. We discuss some of the notable defense mechanisms suggested so far, present the Carlini &

Wagner attack (Carlini and Wagner, 2017b) which was designed to counter them, and conclude this section by describing some more recent, although ultimately unsuccessful, attempts at defending against white-box attacks.

3.1 Preliminaries and Notations

3.1.1 Notations

Throughout this chapter, we use a set of notations to describe the implementation details of adversarial attack and defense methods. We denote the attacked neural network with $f(\cdot)$, use x to signify some valid input to this network, and use Y as the true class label associated with that input. We use N to indicate the number of classes identified by $f(\cdot)$, and $z(x)$ to denote output of the logits layer (the layer that immediately precedes the softmax). We denote the network loss associated with a given pair of input and true class by $J(x, Y)$. And finally, use δ to represent a perturbation pattern that is added to some valid input in order to cause misclassification. We use $x' = x + \delta$ to denote the adversarial example that results from combining the input with the perturbation.

When discussing adversarial attacks, we talk about "crafting" an adversarial example or "finding" it. Although these terms can be used interchangeably, the former term emphasizes the action taken by the attacker in order to create an adversarial example. The latter term, on the other hand, reflects the fact that adversarial examples simply exist within the learning model's input space. As such, the attacker is searching for the relevant adversarial example given some valid input, rather than creating it.

3.1.2 Targeted and Non-targeted Attack Methods

In the context of classification tasks, adversarial attack methods can be broadly divided into two main classes based on the attacker's goal: targeted and non-targeted attacks. In the case of non-targeted attacks, the adversary aims to cause a given input to be misclassified but does not care what class label is assigned to that input. Targeted attacks, on the other hand, aim to manipulate the classifier so that a specific class label C is assigned to that input.

3.1.3 Measuring the Magnitude of Perturbations

In addition to misleading the learning algorithm, adversaries aim to prevent the adversarial examples from being detected. Much of the prior research on adversarial examples dealt with image classification tasks, and hence researchers aimed to

prevent humans from detecting the adversarial perturbation. In this context, three distance metrics have commonly been used as a proxy for human perception:

1. L_0 measures the number of perturbed features (e.g., when the input is an image, the features are pixels).
2. L_2 measures the perturbation's Euclidean norm.
3. L_∞ measures the maximal change to any of the input features.

Perturbation patterns that were designed to minimize the L_0 distance are sometimes referred to as *sparse vector attacks*, while perturbations that minimize the L_2 or L_∞ distance are referred to as *dense vector attacks*.

Applying adversarial attack methods to new content domains requires the identification of a suitable distance metric that correctly represents the defender's abilities. For instance, when designing an attack against a malware detection system, human perception is of lesser importance. Instead, the attacker's goal is to manipulate the malware so that it will not be detected by a classifier, while preserving its malicious nature. When designing an attack against an image recognition system in which humans can also see the input, the perturbations should not be visible to the human eye. Thus, a different distance metric is required in each case.

3.2 The Basic Recipe for a White-Box Attack against a Neural Network

Crafting adversarial examples in white-box conditions generally implies solving two constraints. The attacker aims to find a perturbation δ, which, when added to a valid input vector x, will cause the classifier $f(\cdot)$ to misclassify the perturbed input. This perturbation must be sufficiently small so as to go undetected. In the case of a non-targeted attack, the attacker's goal can be expressed as:

$$\begin{aligned} &f\left(x'\right) \neq Y \\ &s.t.\ \|\delta\|_p \leq \varepsilon \end{aligned} \tag{1}$$

The first constraint in (1) ensures incorrect classification, while the second controls the perturbation magnitude. Here, ε stands for the maximal perturbation radius given some context specific distance metric $\|\cdot\|_p$. Limiting the value of ε aims to prevent the perturbation from being detected. In the case of a targeted attack, a slightly modified version of the first constraint is used:

$$\begin{aligned} &f\left(x'\right) = C \\ &s.t.\ \|\delta\|_p \leq \varepsilon \end{aligned} \tag{2}$$

Under white-box conditions, solving the constraints included in (1) or (2) usually involves:

1. Computing $f(x)$ for some valid input
2. Calculating the loss gradient with respect to the input
3. Performing one or more steps in which the input is perturbed based on the loss gradient calculated in the previous step. In the non-targeted case, perturbation is designed to maximize the loss with respect to the true class, while in the case of targeted attacks, it is designed to minimize the loss with respect to the target class

This process strongly resembles the process of training a neural network. The main difference is that when crafting adversarial examples, the network's weights remain unchanged, and instead the gradient is used for updating the perturbation added to the valid input.

3.3 Early Attack Methods

3.3.1 Fast Gradient Sign Method—FGSM

The first computationally effective approach for crafting adversarial examples against neural networks is commonly known as the Fast Gradient Sign Method (Goodfellow et al., 2014). FGSM was initially designed as a non-targeted method, but it was later extended with a targeted variant. This attack method performs a single iteration of loss and gradient calculation and perturbs each of the input features with a step of a fixed size in the direction of the gradient. More formally, FGSM is expressed as follows:

$$x' = x + \varepsilon \cdot sign\left(\nabla_x J\left(x, Y\right)\right) \tag{3}$$

Perturbing the input in the direction of the gradient increases the network's loss and hence causes the input to be misclassified. The single gradient calculation phase involved in this attack makes it simple to implement and highly efficient in terms of computational needs. Despite its simplicity, this method is quite successful at causing misclassification.

3.3.2 Jacobian Saliency Map Attack—JSMA

Papernot, McDaniel, Jha et al. (2016) proposed the Jacobian Saliency Map Attack method, a greedy iterative algorithm for crafting targeted adversarial examples with a small L_0 perturbation distance. In each iteration, the algorithm computes a saliency map based on the network's Jacobian and then uses this map to choose the two most salient features to perturb. One feature is chosen in order to maximize the loss with respect to the true class, while the other is chosen in order to minimize the loss with respect to the target class. The algorithm stops as soon as the desired target classification is assigned to the perturbed input or after performing a predefined

maximal number of steps. The greedy nature of this algorithm limits the overall number of attack iterations. However, the computation of the complete Jacobian is highly demanding in terms of computational resources, making this attack method impractical for networks with high input dimensionality.

3.3.3 Targeted Gradient Sign Method—TGSM

Based on the FGSM approach, Kurakin et al. (2016a) presented a targeted attack variant referred to as the targeted gradient sign method (TGSM). Instead of maximizing the loss with respect to the true class, the authors aimed to minimize the loss with respect to the target class C:

$$x' = x - \varepsilon \cdot sign\left(\nabla_x J\left(x, C\right)\right) \tag{4}$$

Much like its non-targeted sibling, this attack method is computationally efficient, easy to implement, and remarkably successful at causing targeted misclassification.

3.3.4 Basic Iterative Method—BIM

Another straightforward extension to the FGSM and TGSM is to use a smaller perturbation step and repeat the attack process multiple times. This idea was first suggested by Kurakin et al. (2016b) and is commonly referred to as the basic iterative method (BIM) or the projected gradient method (PGD). Using (3) as a basis, we can formally express the non-targeted BIM variant as follows (note that the targeted variant is expressed by starting from the TGSM equation instead the FGSM one):

$$x_{t+1} = clip\left(x_t + \gamma \cdot sign\left(\nabla_x J\left(x_t, Y\right)\right)\right) \tag{5}$$

In each iteration, the sign of the loss gradient is computed with respect to the output of the previous iteration and the true class. Then, the perturbation step per iteration γ is added to all input features. Finally, the $clip$ operator is used in order to keep each of the input features within its valid range.

The iterative nature of this attack increases the overall attack success rate, while clipping reduces the average perturbation distance. The resulting perturbation is, in most cases, quite refined compared to the single iteration attack methods. By adjusting the maximal number of iterations, one can control the trade-off between computational efficiency and perturbation refinement. Using a few hundred iterations usually provides a good balance point.

3.4 Early Defense Mechanisms

3.4.1 Adversarial Retraining

Adversarial retraining (Goodfellow et al., 2014) is perhaps the most intuitive defense approach against adversarial manipulations. As such, it was included in the publication in which FGSM was proposed. This method is based on the assumption that adversarial examples may exist in areas of the input space that are underrepresented in the training set. With that in mind, the authors suggested augmenting the original training set with adversarial examples that were crafted by the defender and labeled with the true labels of the inputs from which they were derived. Augmentation is performed iteratively, and in each iteration, the defender (1) trains a network so that it is sufficiently accurate with respect to the existing training set, (2) generates adversarial examples against the trained model, and (3) augments the training set with the adversarial examples, along with their true class labels.

In effect, the defender needs to think and act like an attacker in order to defend the network against future attacks. As it turns out, attempting to anticipate all future attack methods is practically impossible; hence, adversarial retraining provides a very limited defense. Networks trained using this simple defense approach demonstrated greater resilience to adversarial manipulations than undefended classifiers. However, this approach suffers from three main shortcomings: (1) the repeated training process is computationally demanding, and hence greatly limits its applicability in the case of large modern networks, (2) adversarial retraining based on weak attack methods does not provide an adequate defense against stronger attacks, and (3) the average perturbation distance does not increase significantly from one training iteration to the next; thus, it is fairly easy to continue to craft adversarial examples against a network that has already undergone several iterations of adversarial retraining.

3.4.2 Defensive Distillation

Defensive distillation (Papernot, McDaniel, Wu et al., 2016) is one of the most notable defense mechanisms against adversarial perturbations suggested so far. It is easy to implement, computationally efficient, applicable for any softmax-based neural network classifier and relies on a solid mathematical foundation. At the time of its discovery, defensive distillation was able to defeat all known attacks, although it was soon defeated by stronger attack methods.

The main goal of the defensive distillation method is to eliminate the loss gradient used in constructing adversarial examples. Its underlying assumption was that without a gradient to follow, attackers would be unable to craft adversarial examples. While this line of thought seems quite logical, later in this chapter we will present attack methods that are not reliant on gradient calculations.

The term distillation refers to the process of training one classifier network using the softmax outputs of another. Hinton et al. (2015) has originally suggested distillation as a mean for reducing the number of neurons needed to perform a given classification task. The aim was to allow the resulting neural network to be used on devices with limited computation resources. Distillation works by first training a "teacher" model to solve the required task and then training a simpler "student" model to predict the teacher's softmax outputs. The distillation process is based on the intuition that by training with the complete set of probability estimates, as opposed to just an indication of the true class, the student model can extract much more knowledge from each training example and therefore be implemented using a smaller number of neurons.

Defensive distillation adapts the original process of distillation described above in order to increase the resilience of the resulting student model to adversarial manipulations. The amount of computational resources used by the student model during inference is of lesser importance in this case, hence there is no need to reduce the model's number of neurons. In order to increase the resilience of the student model, defensive distillation aims to eliminate the model's loss gradient with respect to the input. This is achieved by manipulating the output of the logits layer (the layer that immediately precedes the softmax) by introducing the *distillation temperature*, as explained below. As a result the logit component associated with the most probable class is increased compared to all others logit components. This makes the network appear to be "more certain" of its classification decisions and reduces the value of the loss gradient of the selected class. The input loss gradient is typically decreased by several orders of magnitude until it can no longer be represented by a 32-bit floating point variable. This phenomenon, which was later referred to as *gradient masking* (Papernot, McDaniel, Goodfellow et al., 2017), practically nullifies the gradient observed by the attacker and hence prevents the attacker from crafting adversarial examples.

In the case of defensive distillation the required masking effect is achieved by modifying the softmax formula used by both the teacher and student models slightly and introducing the distillation temperature T as follows:

$$Softmax\left(z\left(x\right), T\right)_i = \frac{e^{z_i/T}}{\sum_{j=0}^{N-1} e^{z_j/T}} \tag{6}$$

Here, N denotes the number of classes identified by the network, $i \in [0, N-1]$ is a specific class label index, and $z\left(x\right) = \left(z_0, z_1, .., z_{N-1}\right)$ is the output of the network's logits layer. When T is equal to one, Eq. (6) reverts back to the original softmax formula, however, as the distillation temperature increases, T eventually becomes larger than any of the logit components z_i, making each of the softmax outputs approach $1/N$. When this happens, softmax assigns equal probabilities to all of the classes, pushing the network into making random classification decisions. During training, the network is therefore required to compensate for the distillation temperature by proportionally increasing the logit values.

When applying defensive distillation, a high distillation temperature (e.g., 30) is used when training the teacher and student models. Then, when the student model is used for inference, the temperature is set back to one. The student's logit values are increased as a result of the training process; however, during inference they are not balanced by the distillation temperature. Eventually, the student's softmax assigns a higher probability to the most likely class. The inherent nonlinearity of the softmax function causes this process to quickly eliminate the loss gradient.

3.5 The Carlini & Wagner Attack—Heralding a Second Generation of Attack Methods

Not long after the publication of defensive distillation, Nicholas Carlini and David Wagner presented a targeted attack method that was specifically designed to overcome it (Carlini and Wagner, 2017b). Their method is commonly referred to as the C&W attack, and in the absence of an official benchmark, it is considered the de facto benchmark for attack success and perturbation refinement. Despite a large number of attempts to defend against it, the C&W attack has so far defeated most, if not all, of the suggested defense mechanisms.

As part of their work, the authors introduced several algorithmic principles which have subsequently been used in many other attack algorithms. Therefore, understanding this attack method is an important step in our journey through the evolution of adversarial attacks and defenses. The C&W attack is based on three main principles:

1. **Ignoring the Softmax Layer**—As discussed in Sect. 3.4.2, the loss gradient propagated from the softmax layer can easily be eliminated. However, such gradient elimination is dependent on the highly nonlinear nature of the softmax function. Therefore, the authors ignored the softmax output and used the output of the logits layer as part of their attack instead.
2. **Reformulation of the Attacker's Goals**—The constraints listed in (2) form a highly non-convex problem, making them difficult to solve. As part of the C&W attack, the authors redefined the optimization problem so that it can be solved using stochastic gradient descent. They introduced a new function $q(\cdot)$, such that $f(x+\delta) = C \Leftrightarrow q(x+\delta) < 0$. This new function was also designed to be monotonic and differentiable.
3. **Tailor-Made Loss Function**—While the early attack methods we surveyed leveraged the neural network's original loss function, Carlini and Wagner were the first to use a dedicated, attack oriented loss function. Detaching the loss used for training a network from the one used for attacking it allows the attacker a great deal of freedom in designing the attack. Carlini and Wagner used this approach in order to allow the attacker a trade-off between attack success and the average perturbation size.

Combining the principles listed above, the authors expressed the task of finding an adversarial example as follows:

$$min\left(\|\delta\|_p + \alpha \cdot q\left(x+\delta\right)\right)$$
$$s.t.\ x+\delta \in [0,1]^n \tag{7}$$

The authors used a standard gradient descent optimizer in order to solve the minimization problem presented in (7). The constant α is used for balancing the effect of the perturbation norm against the value of the function q. Its goal is to ensure that both terms influence the optimization process equally. The result is a powerful set of attacks (one for each commonly used distance metric) for which there are no known defense or detection mechanisms (Fig. 1).

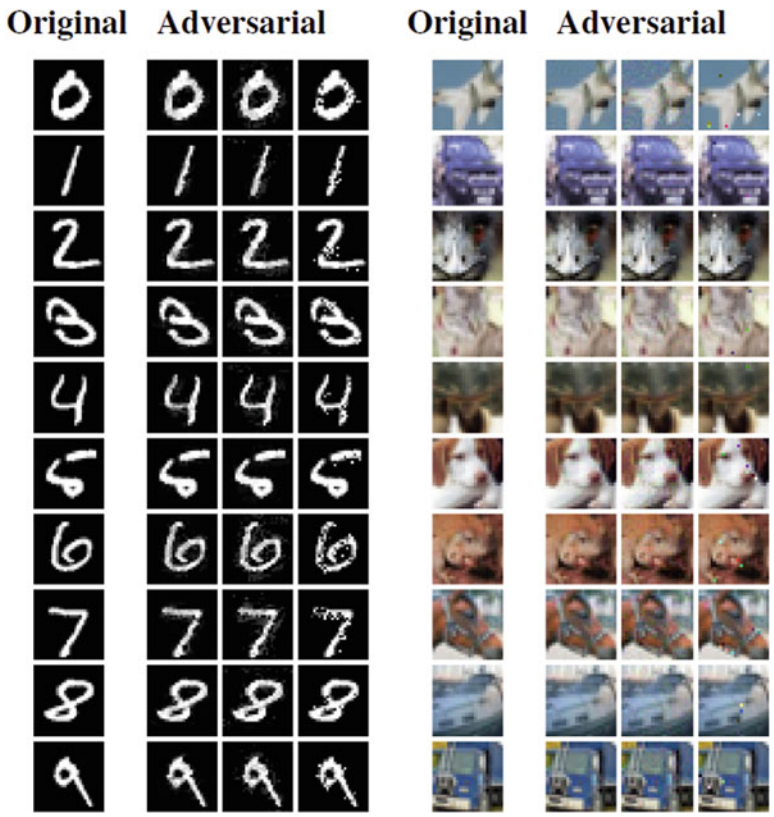


Fig. 1 An illustration of the C&W attack on a defensively distilled network. The four columns on the left illustrate an attack against a MNIST classifier, while the four columns on the right refer to an attack against a CIFAR-10 classifier. In each case, the leftmost column contains the initial image, and the next three columns contain adversarial examples generated by the L_2, L_{inf}, and L_0 attack variants, respectively

3.6 Additional (Unsuccessful) Defense and Detection Methods

3.6.1 GAN-Based Defenses

Based on the assumption that adversarial examples are somehow anomalous or different from "normal" examples, several studies (Ilyas et al., 2017; Lee et al., 2017; Song et al., 2017; Samangouei et al., 2018) suggested using GANs to model the manifold of the true examples and subsequently filtering out adversarial examples. These studies all follow the process presented below:

1. Train a GAN to model the distribution of normal input.
2. Given some input x to be classified, search for a point l in the GAN's latent space such that the generator's output $G(l)$ adequately resembles x:

$$\|x - G(l)\|_p < \varepsilon.$$

3. Assuming such a point is found, use the generator's output $G(l)$ as input to the classifier, and assume $c = f(G(l))$ is the class label assigned to x as well.
4. Alternatively, if such a point l cannot be found, assume that x is an adversarial example.

This algorithm essentially projects x to the generator's latent space, which was not thought to contain adversarial examples. However, Athalye, Carlini et al. (2018) later showed that this assumption is incorrect, and in fact, adversarial examples do exist in the GAN's manifold.

3.6.2 Detecting Adversarial Examples

At some point in the development of attacks and defenses, it became clear that adversarial examples are not easily blocked. That is, making the classifier resilient to such attacks is difficult. This realization led many researchers to focus their efforts on the creation of adversarial example detectors, which operate alongside the classifier network. Various detection methods were suggested, all of which were based on a similar design concept:

1. Some statistical measure is identified for the differentiation of normal inputs from adversarially manipulated inputs.
2. A detector model, which is independent of the main classifier, is trained based on that measure.
3. Inputs to the classifier are first validated using the detector, and only legitimate inputs are actually forwarded to the classifier

The main assumption underlying all of the detection methods proposed is that adversarial examples are anomalous in some way. Given this assumption, it should be possible to identify some statistical measurement that can differentiate adversarial examples from normal examples.

Perhaps the most intuitive detection approach is one in which adversarial examples are identified by analyzing the first several PCA components of the

input. This approach was used in a number of studies (Gong et al., 2017; Grosse et al., 2017; Metzen et al., 2017) in which the authors assumed that the nature of adversarial examples would be reflected in a PCA analysis of the classifier input.

The detection of adversarial examples is in fact a binary classification task. As such, Bayesian uncertainty tests also make a lot of sense. For instance, Feinman et al. (2017), have used a Bayesian uncertainty measure derived from the classifier's dropout layers to construct an adversarial example detector.

Another popular technique involves training a second neural network, independent of the classifier network, for differentiating adversarial examples from normal ones (Bhagoji et al., 2017; X. Li and F. Li, 2017; Hendrycks and Gimpel, 2016). In this case, there is no need to explicitly define a differentiating statistical measure; all that is required is a training set large enough to ensure that the network can be sufficiently trained.

The detection rates reported for some of the methods mentioned earlier in this chapter were very promising; combined with the large variety of detection methods, this led to optimism among the adversarial machine learning research community. Briefly in the attack and defense arms race, it seemed as if detectors provided an effective solution to the problem of adversarial examples. Regrettably, however, Carlini & Wagner were able to prove differently.

In (Carlini and Wagner, 2017a), ten different detection methods were bypassed in white-box conditions. In most cases, this was done by constructing a composite loss function and simultaneously optimizing the perturbation pattern so that it misleads both the detector and the classifier models. Being aware of the ability to simultaneously mislead both the classifier and detector, more recent methods for detecting adversarial perturbations, therefore included some non-differential gradient obfuscation element. Their goal was to make it difficult to evaluate the detector's loss gradient, and thereby thwart the ability to use this gradient for creating adversarial inputs.

Katzir and Elovici (2019) suggested constructing Euclidean spaces based on the activation values of the classifier network's inner layers, then training k-NN classifiers on top of the Euclidean spaces and tracking the classification changes between consecutive layers, in order to differentiate normal input from adversarial examples. The key principle upon which this work is based is that adversarial examples are extremely similar to normal input but end up being classified differently. The similarity of input implies that within the Euclidean spaces associated with the first layers, adversarial examples will be close to the valid input from which they were derived. Similarly, the difference in classification (comparing valid and adversarial input) indicates that within the Euclidean spaces of the last network layers, adversarial examples will be located in proximity to instances of the attack's target class. Together, those two observations lead to some mandatory spatial movement between one Euclidean space and another which serves as the basis for the proposed detection method.

Gradient obfuscation indeed makes it harder to overcome a detection mechanism. However, Athalye et al. (2018) showed it is insufficient. It is not necessary for the attacker to know the exact gradient in order to bypass the defense mechanism. In

fact, all that is required for doing so is the ability to approximate the gradient. Such approximation can be done, for instance, by querying the model for multiple inputs and using the model's outputs for computing the gradient's expectancy $E(J(x, C))$. As shown in the abovementioned research, this approach allows the creation of adversarial examples that can bypass the relevant defense mechanism.

As of this writing, no adversarial example detector has been discovered that can be used in white-box conditions, meaning that if an attacker has knowledge of the detector and its implementation details he/she can craft adversarial examples that can bypass it. While implementing an ensemble of such detectors will increase the resilience of the classifier network, each of those detectors only provide "security through obscurity," and all of them can be breached assuming the adversary is aware of their existence and implementation method.

4 Black-Box Attack Methods

White-box adversarial attacks are extremely interesting from a scientific point of view. They help us understand the limitations of deep learning and shed light on the neural network training process. However, in real life scenarios, where neural networks are used to provide a commercial service, the details of the trained models are considered the service provider's intellectual property. As such, those details are kept secret, making it impossible to launch white-box attacks. The discovery of black-box approaches for generating adversarial examples (Biggio, Corona et al., 2013) allowed attackers, for the first time, to attack a model without knowledge of its implementation details. This discovery almost single-handedly transformed adversarial machine learning from a theoretical problem into a major challenge that concerns academia and industry alike.

4.1 Adversarial Transferability

Adversarial transferability, which was the first black-box approach to be discovered, is one of the most fascinating properties of adversarial examples. As it turns out, adversarial examples are transferable from one model to another, despite differences in the implementation details, model parameters, or even the dataset used for training the two models. This property allows an attacker to manipulate the predictions of a classifier model without any prior knowledge of its structure, parameters, or training set.

Transferability was first discovered in the context of traditional machine learning algorithms (Biggio, Corona et al., 2013) and was quickly adapted to the case of neural networks as well (Szegedy et al., 2013). Finally Papernot et al. (2016) coined the term *transferability*. The attack process utilized in those studies is surprisingly straightforward:

1. Collect a surrogate dataset that is drawn from the same distribution as the one used to train the original model
2. Train a surrogate model based on some best practices or state-of-the-art publications
3. Use one of the known white-box attack methods to craft adversarial examples against the surrogate model
4. Use the adversarial examples created against the surrogate model in order to attack the original model

Despite its simplicity, this attack approach has been shown to be highly effective. Various studies reported attack success rates of 30–70% using this approach. Given that state-of-the art open-source pretrained models and sample datasets are readily available, transferability transformed adversarial machine learning from a theoretical problem into a true concern for machine learning researchers and practitioners.

4.2 Zeroth Order Optimization-Based Attack

Transferability-based black-box attacks operate under the most restrictive attacker model, in which the attacker is assumed to have no prior knowledge of the attacked network, the data used for training it or even query based access to the trained network. A slightly more permissive attacker model is referred to as the *oracle* model. In this case, the attacker can query the attacked network, without knowledge of its internals or training data, in order to classify a limited number of input samples.

The zeroth order optimization (ZOO) attack (P.-Y. Chen et al., 2017) is a recent example of an oracle-based attack method. It assumes that the adversary can query the attacked model in order to obtain the softmax output for a given input sample. To a great extent, this attack can be considered the black-box variant of the C&W attack (Carlini and Wagner, 2017b). Their implementation details are quite similar, with one main difference: in ZOO, the explicit gradient calculation used in the C&W attack is replaced with a zeroth order approximation.

Zeroth order is a derivative-free optimization method that uses oracle queries in order to approximate an objective function $f(\cdot)$. Approximating the gradient of $f(\cdot)$ at a point x is performed using two oracle queries for each of the elements of the input vector:

$$\frac{\partial f\,(x)}{\partial x_i} \approx \frac{f\,(x + h \cdot e_i) - f\,(x - h \cdot e_i)}{2h} \tag{8}$$

Here, h is a small constant, and e_i is a vector. The i^{th} element of e_i is set at one, while all other elements are set at zero. Assuming h is small enough, the gradient can be linearly approximated, as done in (8).

Note that this gradient approximation approach can be used even when the attacked model does not have a well defined gradient at x. As a result the ZOO attack is not restricted to attacking neural network-based classifiers, or even to attacking differentiable models. Given that it does not perform any explicit gradient calculations, this attack can be used against any classification model, differentiable or not.

4.3 The HopSkipJump Attack

The HopSkipJump attack (J. Chen et al., 2019) is another example of an oracle-based attack. The oracle model used in this case is even more restrictive than the one used in the ZOO attack (P.-Y. Chen et al., 2017). Here, the oracle provides a single class label, as opposed to the full softmax output. This attack works by "traversing" the classification boundary of the attacked classifier. The algorithm uses two input samples of different classes: the source input to be perturbed x^* and a sample of the target class $\tilde{x}_0$. Through the iterative process described below, the algorithm follows the boundary line to find a point that is close enough to x^* but is still classified as the target class.

1. At iteration t find the intersection of the classification boundary with the line that connects x^* with $\tilde{x}_t$, using a binary search algorithm. Mark that intersection as x_t.
2. Use queries to the oracle and the Monte Carlo search algorithm in order to calculate an approximated gradient of the boundary line at x_t.
3. Calculate $\tilde{x}_{t+1}$, the target point for the next iteration, by taking a small step from x_t perpendicular to the boundary line: $\tilde{x}_{t+1} = x_t + \varepsilon \cdot \perp \hat{g}$.
4. Repeat steps 1–3 until $\|\tilde{x}_{t+1} - x^*\|_p$ is small enough.

The process is graphically illustrated in Fig. 2.

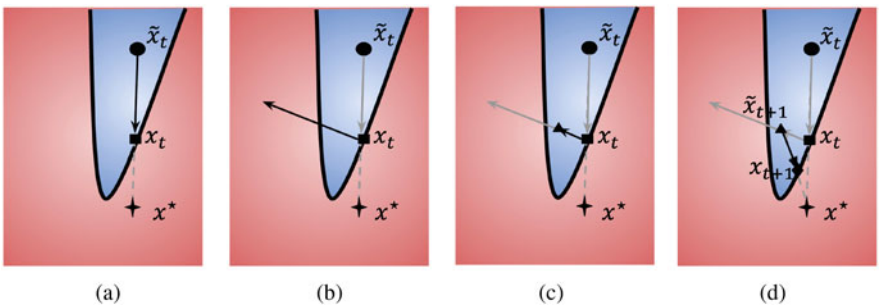


Fig. 2 Graphical illustration of the HopSkipJump attack algorithm (J. Chen et al., 2019): (**a**) finding the boundary point x_t, (**b**) approximating the gradient at x_t, (**c**) taking a small step perpendicular to the boundary, and (**d**) repeating the process to refine the perturbation

Much like the ZOO attack, this attack method can be used with any classification model (not necessarily a neural network). It produces highly refined perturbation patterns and has a very high attack success rate.

5 Real World Adversarial Attacks

Much like the discovery of black-box attack methods, the first demonstrations of adversarial manipulations in the real world dramatically increased interest in adversarial machine learning research. The vast number of machine learning models used in day-to-day applications, combined with the ability to apply adversarial attack methods in the real world created considerable concern. It became clear that malicious actors could leverage those abilities to perpetrate fraud, bypass biometric authentication measures, interfere with the operation of autonomous vehicles, or cause harm in various additional ways.

Kurakin et al. (2016b) were first to demonstrate adversarial manipulations in the real world. In this early work, the authors simply tested the applicability of various methods (FGSM, TGSM, and BIM) in real world scenarios. The experimental setup used was rather straightforward; the authors started with a set of test images, perturbed them using the aforementioned attack methods, printed the resulting images, and finally, used a dedicated smartphone app in order to scan and classify the printed images. Surprisingly, the classifier was subsequently misled in the vast majority of the cases.

Later that year, another research group presented an attack method that was specifically designed to cope with real life constraints (Sharif et al., 2016). The authors demonstrated that a subject could mislead a state-of-the-art facial recognition system by wearing special eyeglasses (glasses with a specially designed frame that contains a perturbation pattern). The frames were printed using a home printer, emphasizing the simplicity and accessibility of the attack. By wearing the eyeglasses, a subject could become invisible to the algorithm, or even impersonate someone else. This effect is illustrated in Fig. 3, which is taken from the original paper.

In addition to breaching a real life facial recognition system, this work had two other notable scientific contributions:

1. The perturbation was confined to a specific part of the input image—the frame of the subject's eyeglasses.
2. The effect of the perturbation pattern remained, even when the frame was shifted slightly with respect to the subject's face.

In order to accommodate the movement of the eyeglasses with respect to the face, the attack algorithm artificially shifted the perturbed region over the input image multiple times, searching for a single perturbation pattern that could mislead the classifier in all cases. Moosavi-Dezfooli et al. (2017) have later extended this concept of optimizing a single perturbation pattern against a variety of inputs in order

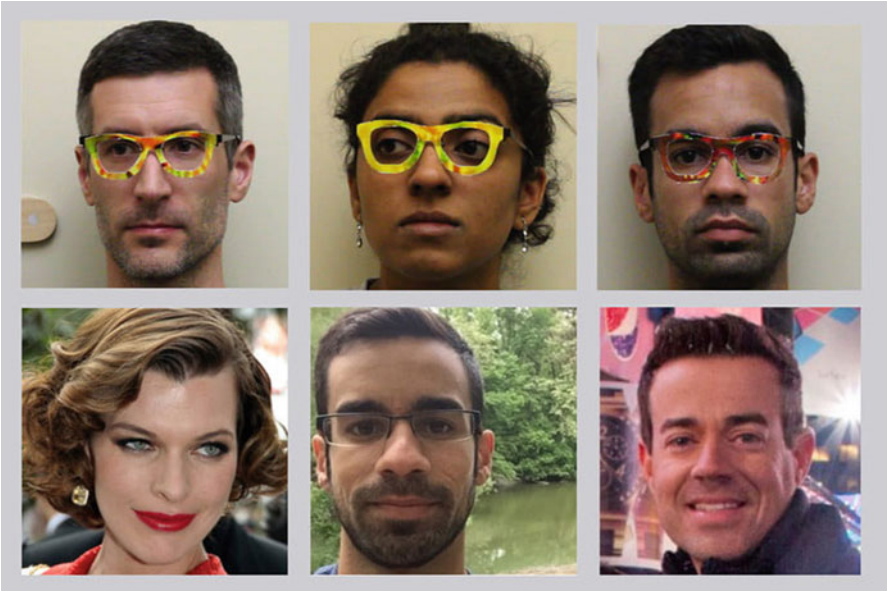


Fig. 3 Accessorize for a Crime (Sharif et al., 2016): Subjects in the top row impersonate the people in the bottom row by wearing eyeglasses with a frame with a specially designed perturbation pattern. Permission to reproduce the figure was granted by the original authors. Bottom right image (by Anthony Quintano; source: https://goo.gl/VfnDct) and bottom left image (by Georges Biard; source: https://goo.gl/GlsWlC.) are under Creative Commons copyrights

to create a "universal" perturbation pattern. A single perturbation that can cause misclassification of a large variety of inputs. This ability dramatically improves upon the attack methods suggested in previous works where the perturbation pattern had to be tailor-made for each and every input sample.

Crafting universal adversarial examples is done iteratively, starting with a single input image and refining the perturbation pattern until it can cause misclassification of multiple inputs at the same time as described below:

1. Initialize the universal perturbation pattern δ to null.
2. Randomly choose an input sample x_i.
3. Augment x_i with the current universal pattern δ and determine whether the classifier can correctly classify $x_i + \delta$.
4. If $x_i + \delta$ is correctly classified, look for δ_i, such that $x_i + \delta + \delta_i$ is misclassified.
5. Assign $\delta = \delta + \delta_i$ and reevaluate all previous input samples.
6. Repeat the process until a predefined number of input samples are misclassified.

Increasing the number of different input samples used to create the universal pattern increases its efficiency but also increases the computation time required for crafting the perturbation. However, empirical tests conducted by the authors showed that using a few hundred examples is sufficient to considerably harm the performance of

a state-of-the-art ImageNet classifier. Furthermore, they showed that such universal adversarial examples are transferable between different models.

Athalye, Engstrom et al. (2017) have further refined the concept of universal perturbations. Here, the authors introduced the Expectation Over Transformation (EOT) framework in order to form adversarial patterns that are resilient to affine transformations of the input. An iterative process, very similar to the one used in (Moosavi-Dezfooli et al., 2017), was used. However, instead of using random inputs from a large training set, the algorithm perturbed multiple views of a single 3D object using a computerized model. Once an incorrect classification was produced for a large enough number of views, the researchers used a 3D printer to create a physical object that is painted with the perturbation pattern. The efficiency of the attack was then evaluated in a real life setting using a video camera and a state-of-the-art object detection network. The final results were impressive—the physical objects created using this approach were incorrectly classified when viewed from a wide range of different angles.

The development of effective universal perturbations also created the tools needed to attack time series or sequence-based classifiers such as Recurrent Neural Networks (RNNs). Such classifiers preserve some internal state vector and use it as part of the classification of new input; universal perturbations allow an attacker to create adversarial inputs that are not dependent on the value of this hidden state variable. Leveraging this concept, Carlini and Wagner (2018) were able to demonstrate an attack against a neural network trained to transcribe an audio recording. By adding a small, barely perceptible noise sample, the authors were able to manipulate the resulting text to their liking.

Collectively, the studies surveyed in this section show that real life systems can easily be manipulated using adversarial examples. Furthermore, addressing real world scenarios with black-box approaches allows such attacks to take place without knowledge of the attacked model's parameters or architecture. In addition to the obvious practical implications, these studies collectively indicate that adversarial examples are much more widespread than initially thought.

6 Theoretical Reasoning and Outstanding Research Questions

A pretty grim picture is revealed when summarizing what we have covered so far in this chapter. Adversarial examples are widespread and are easily and efficiently created using a wide variety of algorithms; targeted adversarial examples give an adversary full control of the output of the attacked model; black-box attack methods allow adversarial examples to be created without prior knowledge of the attacked model or its internals; once created, adversarial examples are transferable between models and can be applied to real world systems, and finally all attempts at blocking, or even identifying adversarial examples have failed.

Taken together, these observations might make one ask: *What makes adversarial examples so difficult to defend against?* This question is perhaps the most interesting and important open question in the adversarial learning domain. Addressing it could assist in defending against adversarial perturbations and could also extend our knowledge about the fundamental nature of neural networks leading to new avenues of research. While a comprehensive answer to this question has yet to be provided, in this section, we explore relevant issues and provide meaningful insights that will contribute to the ability to answer this question.

6.1 Adversarial Examples as a Model Generalization Problem

All machine learning algorithms are based on the assumption that by using a finite training dataset, it is possible to create models that *generalize* well to new, previously unseen data. Achieving optimal global generalization for the entire input space is the main goal of any training process; however, as it is practically impossible to measure the global generalization error directly, the error measured on the training data is typically used as a proxy for the global error. Using this proxy, one can also consider generalization within a confined subspace of the input domain, and by that gain a more fine-grained understanding of the model's performance.

Ideally, the local accuracy (generalization) of a model should be correlated with the density of the training samples in the relevant input subspace. We expect small generalization errors when the spatial density of training samples is high and similarly expect larger errors when the training data is sparse. Adversarial examples, however, challenge our understanding of generalization. On the one hand, they are incorrectly classified despite being located in proximity to a correctly classified input, and it is therefore tempting to think of adversarial examples as evidence of poor model generalization. But on the other hand, adversarial examples do not tend to occur naturally. In fact, despite being susceptible to adversarial perturbations neural networks often generalize extremely well, surpassing human abilities over "normal" input. Treating adversarial examples as evidence of poor generalization hence seems to oversimplify the problem.

Understanding the origins of adversarial examples, therefore, requires that we study the fundamental aspects of generalization in neural networks. Such research might eventually lead to updated definitions of model generalization, as well as adversarial resilience.

In a remarkable work (Zhang et al., 2016), the authors attempted to identify the key to generalization in neural networks. In order to do so, they conducted a simple experiment that led to surprising results. They replaced the labels of the training set with random values and used those random labels in order to train a deep neural network-based classifier. Naturally, the resulting model was useless for predicting the labels of the testing set, but surprisingly, with enough training, it was able to predict the labels of the training set perfectly. Based on their results, the authors concluded that deep neural networks are capable of memorizing their training set

and linked this ability to the large number of model parameters used in deep neural networks.

Follow-up studies (Neyshabur et al., 2017; Kawaguchi et al., 2017) attempted to approximate a neural network's ability to memorize its input using a variety of measures of model complexity. Those studies also attempted to establish guidelines for balancing complexity against the amount of available training data. Relating this back to adversarial examples, this line of research might suggest that adversarial examples are the result of an inability to balance excessive model complexity compared against the limited amount of available training data.

6.2 Adversarial Examples are Inevitable

Recently, an increasing number of studies have been published that deal with the inevitability of adversarial examples. Many of those works established measurements for adversarial resilience and analyzed limitations of those measurements. Referring collectively to many of these works, we can establish the following related, but independent claims:

1. **The inevitability of existence**—Given common definitions of resilience, it is impossible to train a classifier, such that its input domain will be free of adversarial examples.
2. **The inevitability of finding**—Given that adversarial examples exist within the classifier's input space, it is impossible to prevent an adversary from finding them.

Note that the ability to refute either of the claims listed above would suffice for making a classifier resilient to adversarial examples.

6.2.1 Inevitability of Existence

Simon-Gabriel et al. (2019) proved that regardless of their architecture, neural networks become more vulnerable to adversarial examples as the dimensionality of their input increases. More specifically, the vulnerability of neural networks to adversarial examples increases proportionally to the square root of their input dimensionality. Similar to the claims in (Zhang et al., 2016), this work also indicates that adversarial examples are the result of a shortage of training data. However, in this case, vulnerability is associated with the input dimensionality and not the complexity of the underlying neural network. Therefore, as long as the dimensionality of the input remains unchanged, reducing the number of network parameters will not increase its resilience. However, processing high dimensional input is a major strength of neural networks. Reducing input dimensionality therefore makes very little sense from an application point of view.

An alternative explanation for the existence of adversarial examples was provided in (Shafahi et al., 2018). Here, the authors examined the existence of adversarial examples using geometrical tools. They analyzed classifiers of the form $f : [0..1]^n \rightarrow \mathbb{N}$ and placed a formal bound on the minimal perturbation distance ε from any input point required to cause misclassification. This study makes two notable contributions:

1. The minimal perturbation distance decreases as the number of input features increases. This conclusion goes hand in hand with the results reported in (Simon-Gabriel et al., 2019).
2. Their bound is tight enough to allow adversarial examples to go undetected. Using ImageNet classification as an example, they empirically showed that perturbations that adhere to this formal bound are subtle enough to be undetected by humans. This result holds for all three commonly used distance metrics (i.e., L_0, L_2, L_∞)

The same class of classifiers ($f : [0..1]^n \rightarrow \mathbb{N}$) is studied in (Shamir et al., 2019); however, here the authors focused on the L_0 distance metric. By treating the neural network as a piecewise linear transformation from input to output, they too were able to place a formal bound on the perturbation distance. The main contributions of their approach are as follows:

1. The authors do not just prove the existence of adversarial examples in this setting; they also propose an algorithm that can be used to find them.
2. The bound described in this study is tighter (note that only L_0 distances are considered). A numerical by-product of their bound is that perturbing two input features is enough to cause targeted misclassification given any neural network with an input dimensionality of 250 or higher. Modern neural network architectures used for image processing tasks often include many thousands of input features, making such perturbations practically impossible to detect.
3. Finally, this work ties adversarial examples to the piecewise linear nature of neural networks. As a result, the authors conclude that model resilience cannot be improved by modifying the network's architecture or by applying adversarial retraining. Instead, resilience can only be increased if the piecewise linear nature of neural networks is modified.

Approaching adversarial resilience from a different theoretical angle, Gourdeau et al. (2019) suggest that this phenomenon should be studied from a computational complexity point of view. They analyzed binary classifiers that have a Boolean input vector: $f : \{0, 1\}^n \rightarrow \{0, 1\}$. In this context, L_0 is the only applicable distance metric. Their work places formal bounds on the number of bits (input features) that should be changed by an adversary in order to cause misclassification. In order to form those bounds, they used two different definitions for classier resilience based on common notions of adversariality. Both definitions are provided formally within the paper, but we have presented them here informally for the sake of clarity:

1. A classifier should be able to predict the "right" class within a radius of ε from each training sample.
2. When the distance between two input points is smaller than ε, they should be classified similarly by the classifier.

While both definitions seem to intuitively represent our notion of adversarial resilience, both fall short in practice. The former definition builds on the previously discussed concept of classification robustness; however, its evaluation requires the use of an oracle for learning the correct class labels of select inputs. Furthermore, it cannot be extended to cases where input is in the form of $\mathbb{R}^n$. The latter definition eliminates the need for an oracle; however, it cannot be used when the classification problem itself contains input samples with different true class labels that are located in close proximity to one another.

This research makes two key contributions to our understanding of the inevitability of existence of adversarial examples:

1. It proves that, when using any of the previously mentioned definitions, a polynomially large training set is insufficient for ensuring robust classification.
2. It indicates a need to perhaps reconsider, or relax, our definitions for classifier resilience. Indeed, based on the two most commonly used definitions, adversarial examples are inevitable; however, as we have seen, both of those definitions fall short in practice.

6.2.2 Inevitability of Finding

While the abovementioned studies address the issue of inevitability of existence, a recent work (Katzir and Elovici, 2020) explicitly addressed the inevitability of finding. The authors studied the case of softmax-based neural network classifiers, comparing targeted and non-targeted attacks and performing a thorough analysis of the loss gradients. This work formally proves two key findings:

1. The gradients used for crafting targeted and non-targeted attacks are in fact different. Eliminating the gradients used by non-targeted attack methods will not increase the model's resilience against targeted attacks.
2. Targeted attacks use the same gradient that allows neural networks to be trained. This implies that blocking such attacks (preventing the adversary from finding a relevant adversarial example) will come at the cost of losing the network's ability to learn.

Together, these two findings explain why none of the defense mechanisms suggested so far has been able to block targeted attack methods. Furthermore, they suggest the need to rethink the way we train and defend neural network classifiers.

6.3 Additional Open Research Subjects

Apart from the roots for the existence of adversarial examples, which we have addressed above, several additional subjects still require further research:

1. **Providing a solid mathematical definition of a robust classifier**—As we have seen, definitions that stem from our common notion of resilience pose significant practical limitations. There is still a need for a definition of resilience that is broad yet easy to evaluate.
2. **Creating robust real life examples**—Earlier in this chapter, we surveyed various methods for creating adversarial examples in the real world. On average, those methods succeed in the majority of cases. However, in some cases, absolute certainty of the attack's success is required to ensure its applicability in the real world. For instance, consider the case in which a person aims to fool a facial identification system in order to enter a foreign country. In this case, the attacker might not be willing to accept any chance of failure. From a research point of view, we are still investigating ways to predict the success of an attack in real life and to understand the factors that limit this success.
3. **Creating adversarial example detectors valid under white-box conditions**—Many adversarial example detectors have been proposed in recent years, all of which are invalid when the attacker is aware of the detector's existence and implementation details. A detection method that is applicable for the white-box assumption has yet to be discovered.
4. **Providing a universal metric for perturbation distance**—Various distance metrics have been used in order to approximate the defender's ability to detect adversarial examples. When the aim is to fool human perception, as opposed to fooling a computerized detection algorithm, different metrics are used. Each metric fails to represent the defender's abilities in some cases, and choosing the most suitable metric is highly dependent on the specific use case. There is still a need for a better, perhaps universal, metric for adversarial detectability.
5. **Understanding the roots of transferability**—The transferability of adversarial examples from one model to another is known for some time now. However, very little research has examined its origins. More work is required in order to better understand this phenomenon.

7 Summary

Adversarial machine learning has been the focus of a significant amount of research in recent years. Many of the studies performed were aimed at developing effective defense mechanisms against adversarial examples. Despite the large variety of methods proposed, these research efforts have largely failed, making it increasingly clear that *adversarial examples are inevitable*. This understanding has given rise to legitimate concerns regarding the security of machine learning models and revealed

one of the most important outstanding questions in this domain: What makes adversarial examples so difficult to defend against?

In this chapter, we provided an overview of the rapid evolution of the adversarial machine learning domain. We started by examining white-box attack methods and the role played by the input loss gradient in creating adversarial examples. We then reviewed the more realistic, black-box attack scenario where adversarial examples are constructed without knowledge of the attacked model's parameters. Some of the black-box attacks mentioned are applicable when the attacked model has no defined input gradient. We also discussed how adversarial examples can be used in the real world, and we tried to understand the fundamental factors that allow adversarial examples to exist. Finally we have listed additional questions and subjects that still require research focus. In doing so, we aimed to provide readers with the knowledge base needed to begin to explore this fascinating research domain.

Acknowledgements If you want to include acknowledgments of assistance and the like at the end of an individual chapter please use the `acknowledgement` environment—it will automatically be rendered in line with the preferred layout.

References

Athalye, A., Carlini,N., & Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*.

Athalye, A., Engstrom, L., Ilyas, A., & Kwok, K. (2017). Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*.

Bhagoji, A. N., Cullina, D., & Mittal, P. (2017). Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654*.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., . . . Roli, F. (2013). Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 387–402). Springer.

Biggio, B., Nelson, B., & Laskov, P. (2012). Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*.

Biggio, B., Pillai, I., Rota Bulò, S., Ariu, D., Pelillo, M., & Roli, F. (2013). Is data clustering in adversarial settings secure? In *Proceedings of the 2013 ACM workshop on artificial intelligence and security* (pp. 87–98).

Brückner, M., Kanzow, C., & Scheffer, T. (2012). Static prediction games for adversarial learning problems. *Journal of Machine Learning Research*, 13(Sep), 2617–2654.

Carlini, N., & Wagner, D. (2017a). Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security* (pp. 3–14). ACM.

Carlini, N., & Wagner, D. (2017b). Towards evaluating the robustness of neural networks. In 2017 *IEEE symposium on security and privacy (SP)* (pp. 39–57). IEEE.

Carlini, N., & Wagner, D. (2018). Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE security and privacy workshops (SPW)* (pp. 1–7). IEEE.

Chen, J., Jordan, M. I., & Wainwright, M. J. (2019). Hopskipjumpattack: A query-efficient decision-based attack. *arXiv preprint arXiv:1904.02144, 3*.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., & Hsieh, C.-J. (2017). Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In

Proceedings of the 10th ACM workshop on artificial intelligence and security (pp. 15–26). ACM.

Dekel, O., Shamir, O., & Xiao, L. (2010). Learning to classify with missing and corrupted features. *Machine learning, 81(2)*, 149–178.

Feinman, R., Curtin, R. R., Shintre, S., & Gardner, A. B. (2017). Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*.

Gong, Z., Wang, W., & Ku, W.-S. (2017). Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*.

Goodfellow, I., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Gourdeau, P., Kanade, V., Kwiatkowska, M., & Worrell, J. (2019). On the hardness of robust classification. In Advances in neural information processing systems (pp. 7444.7453).

Grosse, K., Manoharan, P., Papernot, N., Backes, M., & McDaniel, P. (2017). On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*.

Hendrycks, D., & Gimpel, K. (2016). Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530*.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., & Tygar, J. D. (2011). Adversarial machine learning. In *Proceedings of the 4th ACM workshop on security and artificial intelligence* (pp. 43–58).

Ilyas, A., Jalal, A., Asteri, E., Daskalakis, C., & Dimakis, A. G. (2017). The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*.

Katzir, Z., & Elovici, Y. (2019). Detecting adversarial perturbations through spatial behavior in activation spaces. In *2019 international joint conference on neural networks (IJCNN)* (pp. 1–9). IEEE.

Katzir, Z., & Elovici, Y. (2020). Gradients cannot be tamed: Behind the impossible paradox of blocking targeted adversarial attacks. *IEEE Transactions on Neural Networks and Learning Systems*.

Kawaguchi, K., Kaelbling, L. P., & Bengio, Y. (2017). Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

Kołcz, A., & Teo, C. H. (2009). Feature weighting for improved classifier robustness. In *Ceas'09: Sixth conference on email and anti-spam*.

Kurakin, A., Goodfellow, I., & Bengio, S. (2016a). Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Kurakin, A., Goodfellow, I., & Bengio, S. (2016b). Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

Lee, H., Han, S., & Lee, J. (2017). Generative adversarial trainer: Defense to adversarial perturbations with GAN. *arXiv preprint arXiv:1705.03387*.

Li, X., & Li, F. (2017). Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE international conference on computer vision* (pp. 5764–5772).

Metzen, J. H., Genewein, T., Fischer, V., & Bischoff, B. (2017). On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., & Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1765–1773).

Neyshabur, B., Bhojanapalli, S., McAllester, D., & Srebro, N. (2017). Exploring generalization in deep learning. In *Advances in neural information processing systems* (pp. 5947–5956).

Papernot, N., McDaniel, P., & Goodfellow, I. (2016). Transferability in machine learning: From phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security* (pp. 506–519). ACM.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016). The limitations of deep learning in adversarial settings. In 2016 *IEEE European symposium on security and privacy (euros&p)* (pp. 372–387). IEEE.

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)* (pp. 582–597). IEEE.

Samangouei, P., Kabkab, M., & Chellappa, R. (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*.

Shafahi, A., Huang, W. R., Studer, C., Feizi, S., & Goldstein, T. (2018). Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*.

Shamir, A., Safran, I., Ronen, E., & Dunkelman, O. (2019). A simple explanation for the existence of adversarial examples with small hamming distance. *arXiv preprint arXiv:1901.10861*.

Sharif, M., Bhagavatula, S., Bauer, L., & Reiter, M. K. (2016). Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 1528–1540). ACM.

Simon-Gabriel, C.-J., Ollivier, Y., Bottou, L., Schölkopf, B., & Lopez-Paz, D. (2019). First-order adversarial vulnerability of neural networks and input dimension. In *International conference on machine learning* (pp. 5809–5817).

Song, Y., Kim, T., Nowozin, S., Ermon, S., & Kushman, N. (2017). PixelDefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Wang, F., Liu, W., & Chawla, S. (2014). On sparse feature attacks in adversarial learning. In *2014 IEEE international conference on data mining* (pp. 1013–1018). IEEE.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Ensembled Transferred Embeddings

Yonatan Hadar and Erez Shmueli

1 Introduction

Deep learning has become a very popular method for text classification in recent years, due to its ability to improve the accuracy of previous state-of-the-art methods on several benchmarks. However, these improvements required hundreds of thousands to millions labeled training examples, which in many cases can be very time consuming and/or expensive to acquire. This challenge has contributed to the emergence and development of multiple active research fields addressing settings with limited labeled data.

Transfer learning is one of the most popular and successfully field of this type, aiming to reduce the need for labeled target data by transferring learned representations from a related model trained on a large labeled dataset (Hedderich, Lange, Adel, Strötgen, & Klakow, 2020). For example, in computer vision, it is common to start the training of a new deep neural network with the weights of a neural network that was pretrained on the large ImageNet dataset (Ruder, 2019). Mikolov, Sutskever, Chen, Corrado, and Dean (2013) suggested Word2Vec, a method for creating unsupervised word embeddings by predicting the surrounding words in a sentence or a document. Word2Vec vectors are commonly used to initialize the first layer of a neural network in many NLP models and showed great benefit in improving model accuracy. Devlin, Chang, Lee, and Toutanova (2018) introduced the BERT model, a Transformer network trained on very large corpus of unlabeled data on the task of masked language modeling and next sentence prediction. BERT has shown state-of-the-art results when finetuned in several tasks such as natural language inference and question answering. However, since most

Y. Hadar (✉) · E. Shmueli
Department of Industrial Engineering, Tel Aviv University, Tel Aviv, Israel
e-mail: shmueli@tau.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_26

aforementioned models were trained with general purpose datasets (e.g., books, news articles and Wikipedia pages), they are expected to be less appropriate in the case of domain specific datasets, especially if the text distribution of the target dataset may differ greatly from that of the corpus they were trained with.

To cope with this limitation, we propose a novel learning framework, Ensembled Transferred Embedding (ETE), which has four main steps: (1) Manually label a small sample dataset (2) Extract embeddings from related large-scale labeled datasets (3) Train transferred models using the extracted transferred embeddings and the labels of the sample dataset (4) Build an ensemble to combine the outputs of the different transferred models into a single prediction

To demonstrate the capabilities of the proposed framework, we evaluate it in the context of item categorization. Item categorization is a machine learning task which aims at classifying e-commerce items, typically represented by textual attributes, to their most suitable category from a predefined set of categories. Many studies have investigated the problem of item categorization as a machine learning text classification task. Here, we focus specifically on item categorization settings in which: (1) Item descriptions are relatively short and noisy. (2) Labeled data for the target dataset is unavailable. Such settings entail that deep learning techniques cannot be applied directly on the target dataset, and that using transfer learning from general purpose models may not be optimal.

Extensive evaluation that we conducted, using a large-scale real-world invoice dataset provided to us by PayPal, shows that the proposed ETE framework significantly outperforms all other considered traditional (e.g., TF-IDF) as well as state-of-the-art (e.g., methods based on general purpose pretrained models such as BERT) item categorization methods.

The majority of the content appearing in this chapter is based on our previous journal publication (Hadar & Shmueli, 2021a).

The rest of this chapter is structured as follows. In Sect. 2, we describe the proposed ETE learning framework. In Sect. 3, we detail how the ETE framework can be applied to the item categorization task. Section 4 discusses the experimental setting and the results. Section 5 summarizes this chapter and suggests directions for future work.

2 The ETE Framework

In this section, we propose the Ensembled Transferred Embedding (ETE) learning framework. The ETE framework first manually labels a small sample of instances from the target dataset and generates embeddings from related large-scale datasets. Then, each set of generated embeddings and the labeled instances of the target dataset are used to train a model that is tailored to the target task. Finally the set of trained models are combined into a single ensemble to provide enhanced performance. This motivation behind this approach is twofold: (1) Using related (or "transferred") embeddings as opposed to general purpose embeddings makes

the resulting model more tailored to the target task. (2) Generating the embeddings from related datasets bypasses the need to label a large portion of the target dataset.

Our Ensembled Transferred Embeddings framework is composed of four main steps (see Fig. 1):

- Step 1: Sample Dataset. In this step, we generate a relatively small-scale manually labeled dataset. This may be achieved by randomly sampling a relatively small number of instances from the large-scale unlabeled target dataset, and manually label these instances. Labeling can be done by using domain experts, crowdsourcing, heuristics, or a combination of the above.
- Step 2: Transferred Embeddings. In this step, we extract embeddings from related large-scale labeled datasets. To achieve this goal, we first need to identify related datasets. Such datasets are large-scale labeled datasets from a domain or task, similar in nature to that of the target dataset. Such datasets may include, for example, a publicly available dataset with a similar task from a different domain, a dataset that was gathered for a different task on the same domain, or even a self-supervised task on the target dataset. After the related datasets were obtained, a deep neural network is trained on them to obtain an "embedding network." Finally, the sample dataset is provided as input to the embedding network to generate "transferred embeddings," in a manner similar to (Sharif Razavian, Azizpour, Sullivan, & Carlsson, 2014) and (Kiros et al., 2015).
- Step 3: Transferred Models. In this step, we train models using the extracted transferred embeddings and the labels of the sample dataset. More specifically, for each set of transferred embeddings and their corresponding labels (from the sample dataset), we apply a supervised machine learning algorithm to obtain a "transferred model."
- Step 4: Ensemble. In this step, a meta model for combining the outputs of the various transferred models is built. This can be obtained in several ways, for example, by using domain knowledge, applying a voting rule such as plurality (see the work by Werbin-Ofir, Dery, & Shmueli (2019) for more details), or by training a meta model to learn the right combination method.

Given a new data instance we first use the trained embedding networks to extract transferred embeddings. We then apply the transferred models on the extracted transferred embeddings. Finally, we combine the outputs of the various transferred models using the ensemble to decide on the predicted category.

3 Applying ETE to the Item Categorization Task

In this section, we describe the required details for applying the ETE framework to our item categorization setting. Specifically, we elaborate on the datasets that were used; how the sample dataset was extracted and labeled; how the transferred

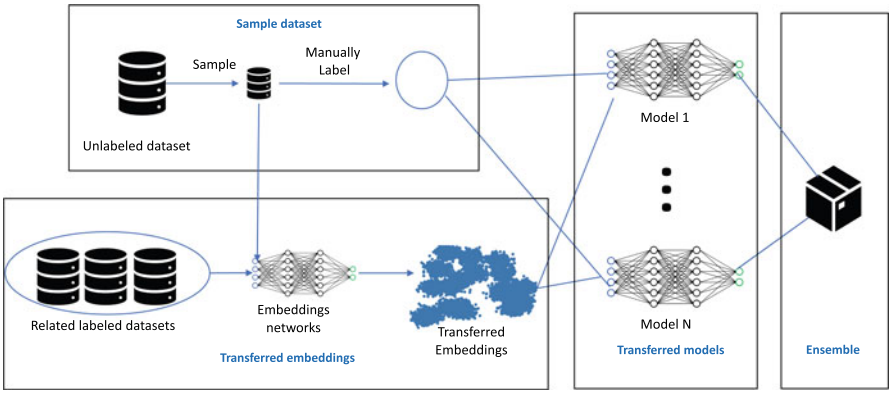


Fig. 1 High-level architecture of the ETE framework

embeddings were obtained; how the transferred models were trained; and how the ensemble was built.

3.1 Datasets

3.1.1 Invoice Dataset

Our main dataset contains 3,461,897 invoices from PayPal's P2P transaction service, sampled and provided to us by PayPal's team. The invoices were sampled randomly from all invoices, having a seller from the USA, and a creation date between September 15–October 13, 2018. Each transaction in our dataset contains the following fields: item name, item description, price, quantity of items, Buyer ID, seller ID, and seller's industry. Table 1 presents a few representative examples of such invoices.

The item name and description fields are free text fields that are filled by the seller without any constraint nor validation. Thus they are typically very short and noisy[1] (see, for example, invoices 1 and 3 in Table 1), and sometimes not informative at all with regard to the sold item nor its category (see, for example, invoices 5 and 6 in Table 1). It should also be noted that item description is an optional field and therefore is not necessarily filled by the seller. In fact, it contains a value only in 24% of the invoices in our dataset. Figure 2 presents a histogram of the number of words per item (after concatenating the item name and description fields).

[1] We use the term noisy to describe user generated text that typically contain grammatical errors, nonstandard spellings, abbreviations, etc., as previously done with tweets on Twitter, (Baldwin et al., 2015).

Table 1 Example of six PayPal's invoices and their attributes

Invoice ID	Item name	Item description	Creation date	Price	Seller's industry	Seller ID	Buyer ID
1	Orange and red bralette large		2018-09-18	23.15	fashion	222	42
2	Voice over recording	Recorded the part of Michael the CEO for the entirety of the project.	2018-10-12	1000	Media	15	765
3	OS leggings		2018-09-07	38	Fashion	12	899
4	Transport to north Ireland for eBay pellet hose		2018-10-01	32.62	Auto-parts	65	78
5	Product		2018-09-18	81	Photography	132	65
6	Bunnies	1/2 yard	2017-11-12	27.75	Arts-n-craft	2	555

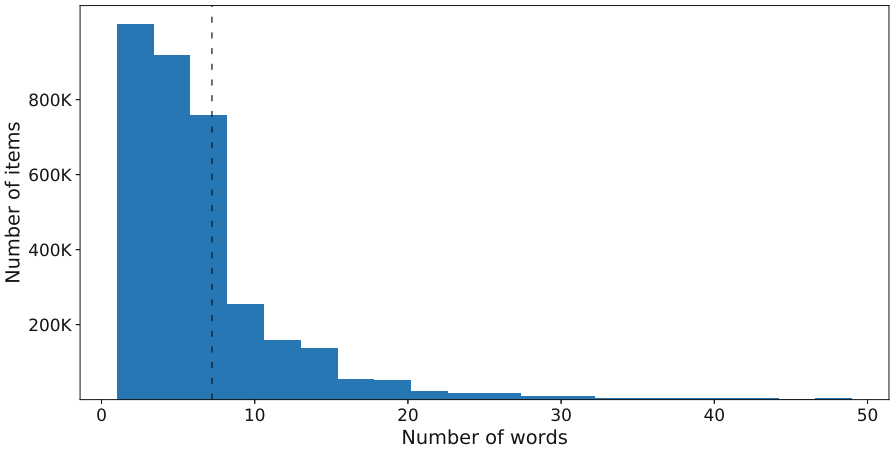


Fig. 2 Histogram of the number of words per item description in the invoice dataset. The vertical dashed line represents the mean number of words per item description

The seller's industry field represents a classification of the seller itself (i.e., not a specific invoice/item) into a single category out of 40 predefined categories (such as fashion, jewelry, accounting, etc.). (The goal of this chapter is to classify the item sold in an invoice into one of these 40 categories.) This field is filled in the vast majority of cases automatically using PayPal's proprietary algorithm. This field is highly imbalanced, where the most popular category (fashion) contains 24% of the invoices, and the 10 most popular categories contain 78% of the invoices. A histogram of the various categories is presented in Fig. 3

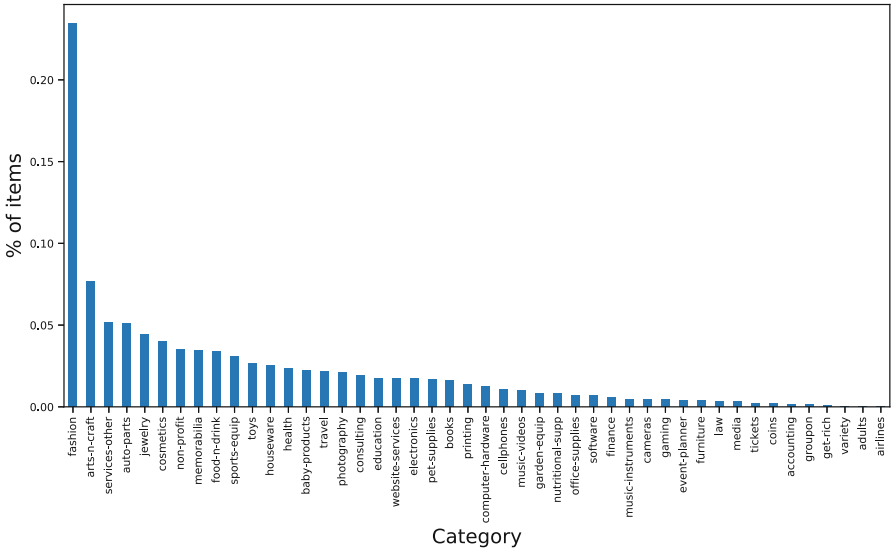


Fig. 3 The distribution of items by seller's industry in the invoice dataset

It should be noted that before providing us the invoices, PayPal performed an initial filtering procedure which originally involved roughly 8 million invoices, and filtered-out invoices with the following properties:

- Item description is shorter than 3 symbols.
- Item description contains a single token which is one of the following: "invoice," "order," and "unknown."
- Invoice items with duplicate values in the text attributes were kept once.

The 3,461,897 invoices analyzed in this study are those remaining after the initial filtering procedure.

3.1.2 eBay Dataset

This dataset contains 2,997,571 items that were extracted using eBay's public API. The items were sampled randomly from all items on the eBay US website (eBay.com), having a creation date between September 1–October 26, 2018. We only used items that were sold at least once. Each extracted item contains the following fields: item description, country, price, creation date, and category. Table 2 presents a few representative examples of such items.

The item description field in this dataset is significantly longer and less noisy than that of the invoices dataset (see, for example, record 1 in Table 2). This is in part since it is used to retrieve items in eBay's item search. Moreover, eBay encourages sellers to augment their item descriptions with relevant keywords, which

Table 2 Example of five eBay items and their attributes

Record ID	Item description	Country	Price	Creation date	Category
1	"VINTAGE STERLING SILVER 20"" long FINE ROPE LINK NECKLACE CHAIN - 3g!"	GB	24.38	2018/10/15	jewelry
2	Off White Red iPhone X SE 5 6 7 8 S Plus Off White iPhone Case [For Apple iPhone 8 Plus]	ID	22.98	2018/09/20	Cellphones
3	Superhero Smash Hands Gloves Ironman spiderman Hulk The Avengers 1 Pair new [hulk]	C2	11.16	2018/09/04	Toy
4	SWIFTLET Tempered Glass Screen Protector for iPad 2 3 4 5 6 Air Mini Pro 9.7 [iPad Air 1/2]	US	13.24	2018/09/14	Arts-n-craft
5	Real Premium Tempered Glass Film Screen Protector for Apple iPad 1/ 2 / 3 / 4	SG	25.07	2018/10/14	Furniture

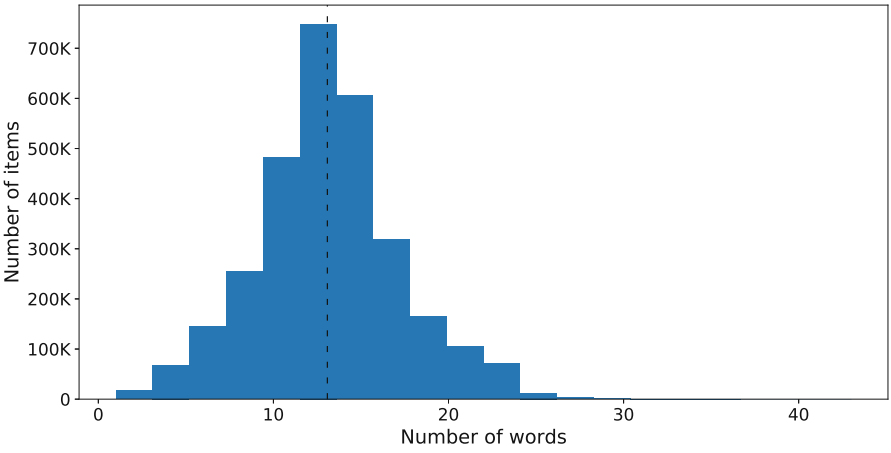


Fig. 4 Histogram of the number of words per item description in the eBay dataset. The vertical dashed line represents the mean number of words per item description

makes the description of two related items quite similar. For example, records 4 and 5 in Table 2 both share the keywords Tempered Glass, Screen Protector, and iPad. Figure 4 presents a histogram of the number of words per item description.

The category field here classifies an item into one of eBay's predefined set of categories. It is important to note that the set of eBay categories is different from the set of PayPal categories. First, the set of eBay categories contains thousands of categories while the set of PayPal categories contains only 40 categories. Second, eBay categories are limited to goods (i.e., physical items), whereas PayPal categories include also services. To overcome this difference, eBay categories were mapped to their corresponding PayPal categories (33 out of the 40 PayPal categories

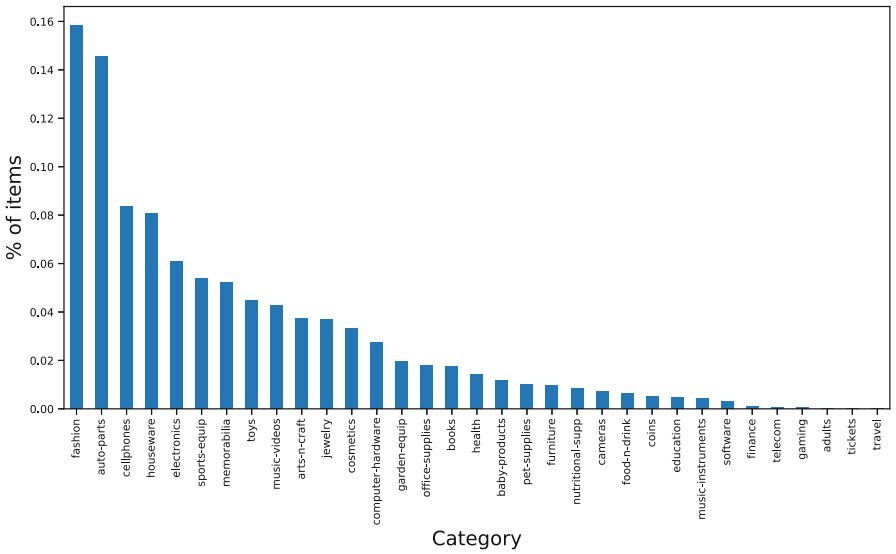


Fig. 5 The distribution of items by category in the eBay dataset

that represent goods) using a set of rules determined manually by PayPal's domain experts. A histogram of the resulting set of categories is presented in Fig. 5

3.2 Sample Dataset—Labeling Using MTurk

Our sample dataset was obtained by a uniform sampling of 1970 instances[2] from the invoices dataset and manual labeling of these instances using Amazon Mechanical Turk (MTurk) as we proceed to explain.

Amazon Mechanical Turk is a marketplace for crowdsourcing of Human Intelligence Tasks (HITs) that is often used to generate labels for supervised machine learning tasks.

Each of our 197 MTurk tasks was composed of labeling a set of 10 instances (1970 instances in total) and was sent to 5 different workers (Turkers). Each task started by providing a general description of the study, followed by a detailed explanation of how the questions should be approached, and a detailed description for each of the 40 categories (including examples of items that fit that category).

Then, the Turker was asked to answer the task. For each of the 10 instances appearing in the task, we provided the Turkers with the item name and description and asked them to answer the following three questions:

[2] The specific number of 1970 instances was chosen to fit our budget constraint of 200 USD.

1. Can you understand what the sold item is? [yes/no]
2. Can you choose the most suitable category for the sold item? [from a list of provided categories]
3. Were you able to adequately categorize this item? [yes/no with explanation]

An example of such a task is presented in Fig. 6.

It is important to note again that the set of PayPal categories includes 40 categories and requesting the Turker to go through a list of 40 categories to choose the one that best fits the item description is a tedious task. Indeed, in preliminary experiments that we conducted, we found that providing all 40 categories as options led to a high level of disagreement between Turkers and to a high level of bias towards the categories shown at the beginning of the list.

To overcome this issue, we decided to split the list of categories into 3 parts: (1) Most popular—the 10 most popular categories, according to the seller's industry attribute, presented in alphabetical order (2) Computer guessed—the 5 most probable categories (that are not among the 10 most popular categories) according to some baseline model (Logistic regression on TF-IDF representation with industry as the label) (3) Other categories—the rest of the 40 categories Clearly, while addressing the bias mentioned above, this design could lead to a different type of bias, by which items from the most popular or computer guessed categories would have higher likelihood to be chosen. Nevertheless, we believe that such a bias is minor since Turkers still chose categories from the "other categories" list in a non-negligible number of cases, and the overall level of agreement between Turkers increased considerably.

In order to further help Turkers categorize the items in a fast and accurate way, we provided them a hyperlink to a Google search of the item name, and for each category in the list of categories, we provided a tooltip that included the category's description (the short description of the category that also appeared in the instructions page).

After completing the data collection stage, we reviewed the data to remove "lazy Turkers" (i.e., Turkers that have not spent enough time and effort to answer the tasks). We did so by first calculating the level of agreement for each instance (i.e., the maximum number of Turkers that have assigned the same category to that instance). We then identified instances with a level of agreement between 3 to 5. Finally, we removed Turkers (and their answers) that agreed with the most-agreed category in less than 20% of these instances (and all of their answers). Since we used Turkers with a master title (required additional payment to MTurk), we found only two such "lazy Turkers." After their removal, we sent all of their tasks to MTurk to obtain new and valid answers.

Finally, we used the resulting data to label the instances. If 3–5 Turkers answered "no" to the first question of a given instance, that instance was labeled as uninformative (15% of the instances). If 3–5 Turkers agreed on the category, this category was assigned as the label for that instance (75% of the instances). Finally, the rest of the instances were reviewed manually by a domain expert in order to choose the best fitting category (10% of the instances).

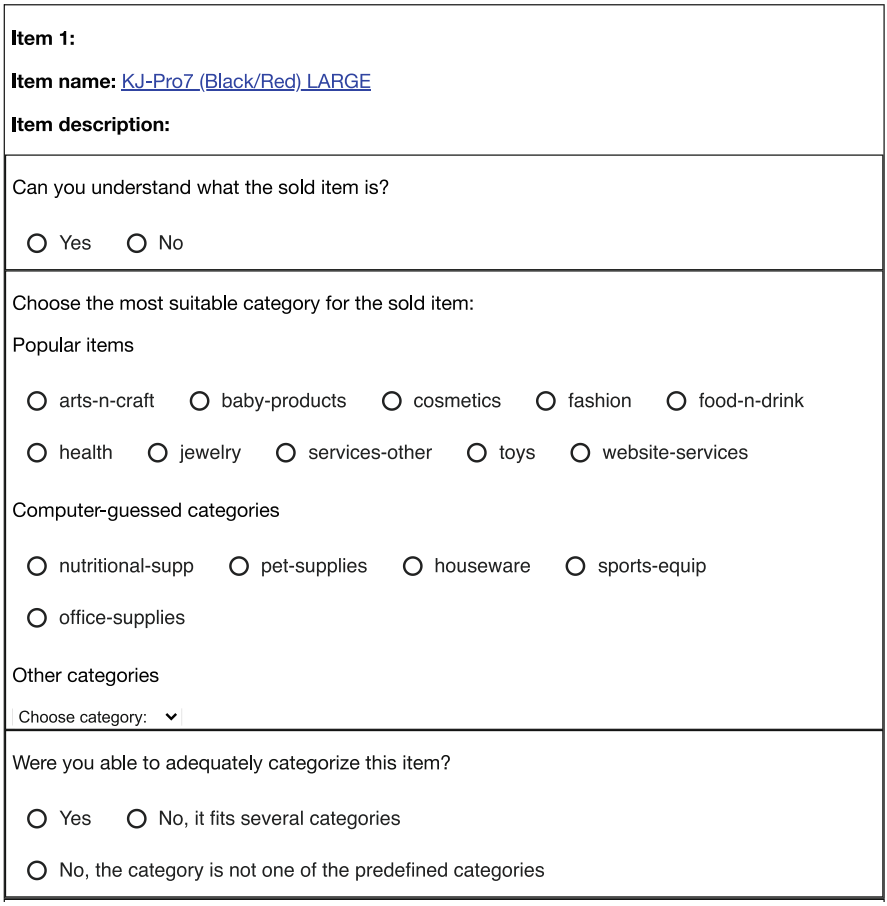


Fig. 6 An example of a single instance to be categorized as appeared in the Mechanical Turk task

3.3 Transferred Embeddings—Using Seller's Industry Attribute and eBay Dataset

We use three types of transferred embeddings (as illustrated in Fig. 7):

- **Industry embedding**.
  Recall that we have access to a large number of invoices, but none of these invoices is labeled (i.e., the right item category for each invoice is unknown). Fortunately, recall that each invoice is associated with the seller's industry category. While the seller's industry category is clearly related to the item category (fashion website will mostly sell fashion items), the two categories are not identical for two main reasons: (1) Sellers usually sell more than one

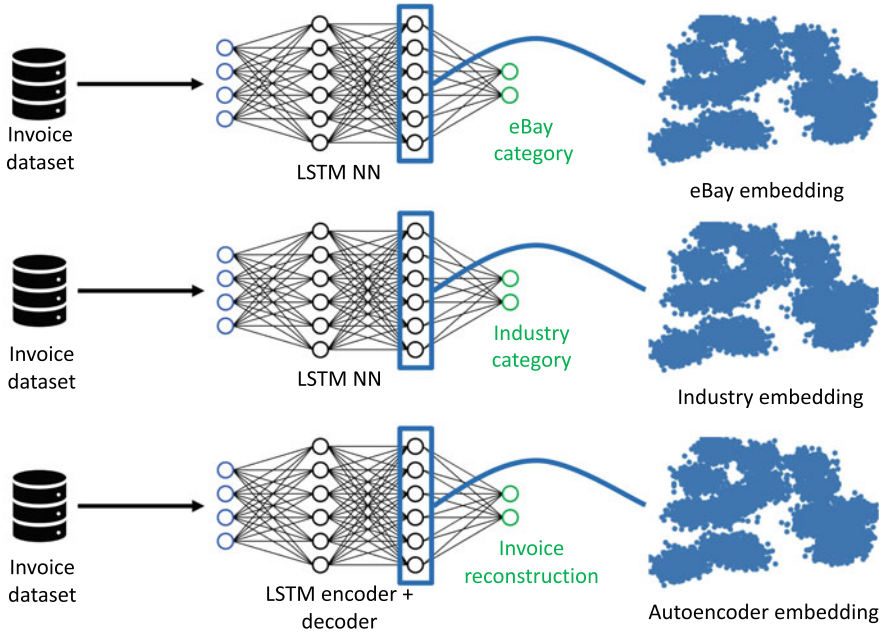


Fig. 7 The process for generating transferred embeddings in our setting

category of items (e.g., a fashion website can also sell jewelry or baby clothes). (2) The seller's industry category itself may be wrong (since it was generated by an automated algorithm). To further support this claim, an analysis of the MTurk dataset shows that the seller's industry category matches the item category in 48% of the cases only. To conclude, since the seller's industry category can be seen as a noisy label for the item category, we train an LSTM neural network to predict the seller's industry category from the item description and use the last layer before SoftMax as the industry embedding (see more details below). It is important to note that due to the differences between goods and services categories we decided to split the industry embedding into two types of embeddings: a goods embedding and a services embedding. Each of the two embeddings was generated by training the LSTM neural network only on part of the invoices data (either those that have a seller's industry category associated with goods or those that have a seller's industry category associated with services).

- **eBay embedding**.
  In contrast to the invoice dataset, the eBay dataset has reliable item category labels, but it differs from the invoice dataset in two ways. First, the eBay dataset contains only a subset of the potential invoice categories since eBay sells goods while our invoices, which are the result of P2P transactions, contain also services. Second, the item description in eBay is longer, more standardized, and

less noisy than the item description in the invoice dataset, since it is aimed to be found in the eBay search. To conclude, since in both cases we aim at predicting the item category from the item description, we train an LSTM neural network to predict the eBay's item category from the eBay's item description and use the last layer before SoftMax as the eBay embedding (see more details below).

- **Autoencoder embedding**.
  The last type of embedding is an embedding trained using a self-supervised Autoencoder. An Autoencoder is a model that is trained to reconstruct its own input. We use a similar LSTM architecture to the two neural networks mentioned above for the encoder (without softmax) and a mirror architecture with a fully connected layer in the end for the decoder.

The three LSTM deep neural networks mentioned above share the same architecture and hyper-parameters. We chose to use LSTM deep neural networks due to their successful implementation in previous works on item categorization tasks (Krishnan & Amarthaluri, 2019; Li, Kok, & Tan, 2018). The chosen architecture and hyper-parameters are detailed in Sect. 4. The rationale for using the exact same architecture and hyper-parameters in the three networks was merely a matter of simplicity.[3]

3.4 Transferred Models—Using a Fully Connected Neural Network

After generating the transferred embeddings described above, we turn to train "transferred models," each trained using a different transferred embedding and the Mturk dataset. More specifically, given a train set of instances extracted from the MTurk dataset, for each transferred embedding (separately), we generate a new training set. The features of the new training set are generated by applying the transferred embedding on the features of the original MTurk instance, and the target value is simply copied as is from the original MTurk instance. The resulting set of instances is then used to train a transferred model. For this purpose we use a relatively small fully connected neural network with 1 layer of 100 hidden units and a softmax layer.

3.5 Ensemble—Stacking Models

To combine the results of the four transferred models, we use a common ensemble technique called model stacking (Wolpert, 1992). More specifically, given a train set of instances extracted from the MTurk dataset, we apply each of the transferred

[3] Our goal here was to demonstrate the advantages of the ETE framework on a large-scale real-world problem, rather than pursuing the best possible accuracy.

models to obtain a prediction (a features vector containing a single probability value for each of the 40 possible categories). Next, a new instance is produced by unifying the predictions into a single features vector and copying the target value as is from the original MTurk instance. Finally, the resulting instances are used to train a Logistic Regression meta model. This process allows our stacking meta model to learn the strengths and weaknesses of each transferred model and consequently lead to improved predictions.

4 Evaluation

In this section we report the extensive evaluation that we conducted. We begin by describing the experimental setting in Sect. 4.1, followed by the results in Sect. 4.2. The source code used for the proposed framework and its evaluation can be found in (Hadar & Shmueli, 2021b).

4.1 Experimental Setting

The ETE method was evaluated by comparing its classification performance to that of seven other benchmark methods (see Sect. 4.1.1) over the labeled MTurk dataset (see Sect. 4.1.2). We applied a stratified 10-fold cross validation evaluation scheme and used accuracy and weighted F1 score[4] as measures for classification quality.

4.1.1 Compared Methods

We compare our method to four common item categorization methods and three methods based on transferred embeddings.

- **Majority:** Our first method is taking the most frequent category in our dataset (fashion) as the predicted class.
- **TF-IDF:** Our first baseline model is a regularized logistic regression model trained on TF-IDF with unigrams and bi-grams, similar to (Kozareva, 2015). The TF-IDF was built using the 3500 most popular terms in the invoice dataset.
- **Average GloVe:** A fully connected neural network trained on average pre-trained GloVe word embedding as input vectors, similar to (Kozareva, 2015).
- **BERT:** A fully connected neural network trained on embeddings extracted from BERT base uncased (Devlin et al., 2018).

[4] The harmonic mean of precision and recall of each class weighted by the class proportion in the data.

- **Autoencoder:** A fully connected neural network trained on transferred embedding extracted from a self-supervised autoencoder (Erhan et al., 2010) trained on the invoice dataset as described in Sect. 3.3.
- **eBay embedding:** A fully connected neural network trained on the eBay transferred embedding as described in Sect. 3.3.
- **Industry embedding:** A fully connected neural network trained on the Industry transferred embedding as described in Sect. 3.3.

4.1.2 Dataset and Pre-processing

All of our experiments were conducted using the MTurk dataset. The two text attributes, item name and item description, were then concatenated to a single text attribute. The text attribute was then pre-processed by applying very basic text operations, such as tokenization, lower-casing, and removing non alpha-numeric symbols.

4.1.3 Hyper-parameter Tuning

Recall that all three LSTM deep neural networks used for extracting embeddings shared the same architecture. The chosen architecture and hyper-parameters were tuned using a grid search on the following hyper parameters: Number of LSTM layers (1–3), size of hidden dimension for the LSTM layers (100,200,400), size of hidden dimension for the fully connected layer (10,30,50,100) and dropout rates (0.1,0.2,0.3,0.5). The chosen architecture and hyper-parameters are shown in Table 3. We used Adam optimizer in all runs with a learning rate of 0.001.

4.2 Results

First, we computed the average classification accuracy and Weighted F1 score (over the 10 folds) for each one of the eight compared methods (see Fig. 8). As

Table 3 Chosen architecture and hyper-parameters for the LSTM deep neural networks

Layer number	Layer	Size	Parameters
1	Embedding	300	Frozen pretrained GloVe, max length= 15
2	Spatial dropout		Dropout rate=0.3
3	LSTM	200	Return sequence=True
4	LSTM	100	
5	Dropout		Dropout rate=0.2
6	Fully connected	30	Activation=relu
7	Fully connected	Number of classes	Activation=Softmax

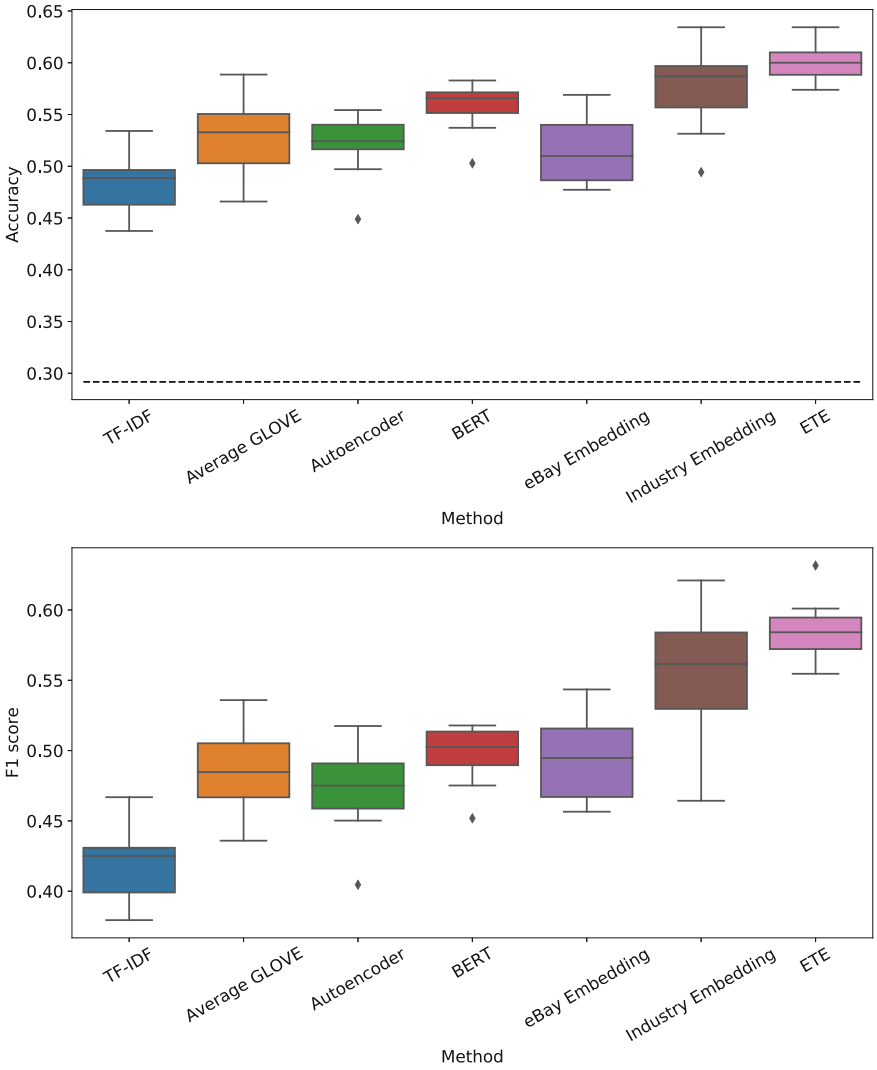


Fig. 8 A comparison of the different item categorization methods in terms of: Accuracy (top) and Weighted F1 score (bottom). The horizontal dashed line in the top subfigure represents the accuracy of the Majority method

expected, all six embedding-based methods performed better than the basic TF-IDF methods. Moreover, we see that the Industry embedding method (which is based on transferred embedding) was able to outperform BERT which is the state-of-the-art general purpose embedding method. We believe that this happens due to the unique characteristics of the text used in our setting—item descriptions are short, noisy and are domain specific. These characteristics make the text distribution in our setting

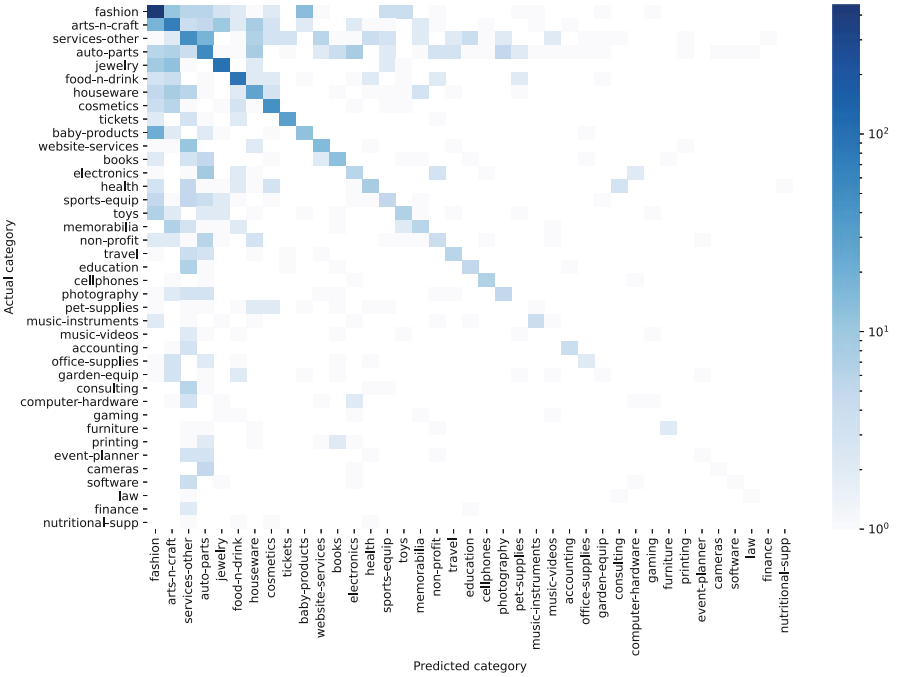


Fig. 9 The classification confusion matrix

different than that of the text used to train BERT, making BERT less suitable for our setting. Finally, we see that ETE, which ensembles several transferred embeddings, performs the best in terms of classification accuracy and Weighted F1 score and has a low variance across folds compared to other methods.

To further support our findings above, we performed statistical significance tests. First, we performed an ANOVA analysis in order to reject the null-hypothesis that all compared methods performed the same. Second, we performed the Tukey post hoc test to perform a pairwise comparison between the various methods. With a confidence level of 95%, we rejected the null-hypothesis that all methods performed the same. Further applying the Tukey post hoc test supported our finding that ETE performed significantly better than all other methods.

To better understand the cases in which our model had mistaken, we computed the classification confusion matrix (see Fig. 9). Note that categories on the two axes are sorted according to their overall popularity. As expected, most of the model's classifications were successful, resulting in a relatively dark diagonal line. Some worth noting confusions are between the fashion category and the baby-products category, and between the jewelry category and arts-n-craft category. Although our evaluation considered such confusions exactly the same as a confusion between the electronics category and the food-n-drink category, it is clear that the former confusions are considerably more tolerable.

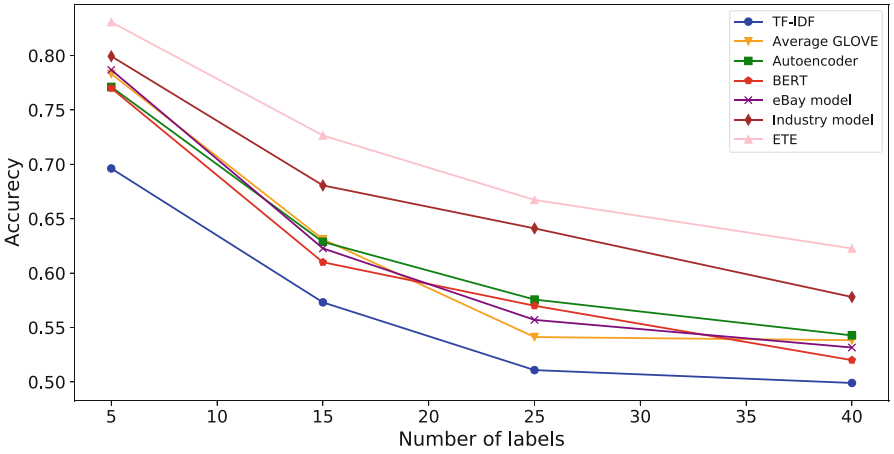


Fig. 10 Accuracy as a function of the number of categories considered

Since the number of PayPal categories is relatively large, the likelihood of our model (and in fact any model) to confuse between them is high. This is especially true for rare categories for which the model had a small number of instances to train with. In Fig. 10, we demonstrate the impact of restricting the number of categories to the top X most common categories (for X ranging between 5 and 40) and ignoring all instances that belong to the other categories, on the accuracy of the various compared methods. As expected, we see that all methods performed considerably better when restricted to a lower number of categories. Moreover, we see that the proposed ETE method outperforms the other method in all considered cases.

Finally, we also tested the effect of using different machine learning classification algorithms for training the transferred models as well as for training the ensemble model (see Fig. 11). The box plot on the top of the figure shows the effect of using different classifiers for training the transferred models (while keeping the ensemble classifier fixed—using Logistic Regression). As can be seen from the figure, using a Neural Network classifier slightly outperformed the others, but this difference was not found to be statistically significant. More specifically, when applying an ANOVA test with a confidence level of 95%, we rejected the null-hypothesis that all models performed the same. Further applying the Tukey post hoc test, we found that Logistic Regression, Random Forest, and Neural Network performed significantly better than Gradient Boosting, but the difference between the three models was not found to be significant.

The box plot on the bottom shows the effect of using different classifiers for training the ensemble model (while keeping the classifier of the transferred models fixed—using Neural Network). As can be seen from figure, using a Logistic Regression classifier slightly outperformed the others, but this difference was not found to be statistically significant. More specifically, when applying an ANOVA test with a confidence level of 95%, we could not reject the null-hypothesis that all models performed the same.

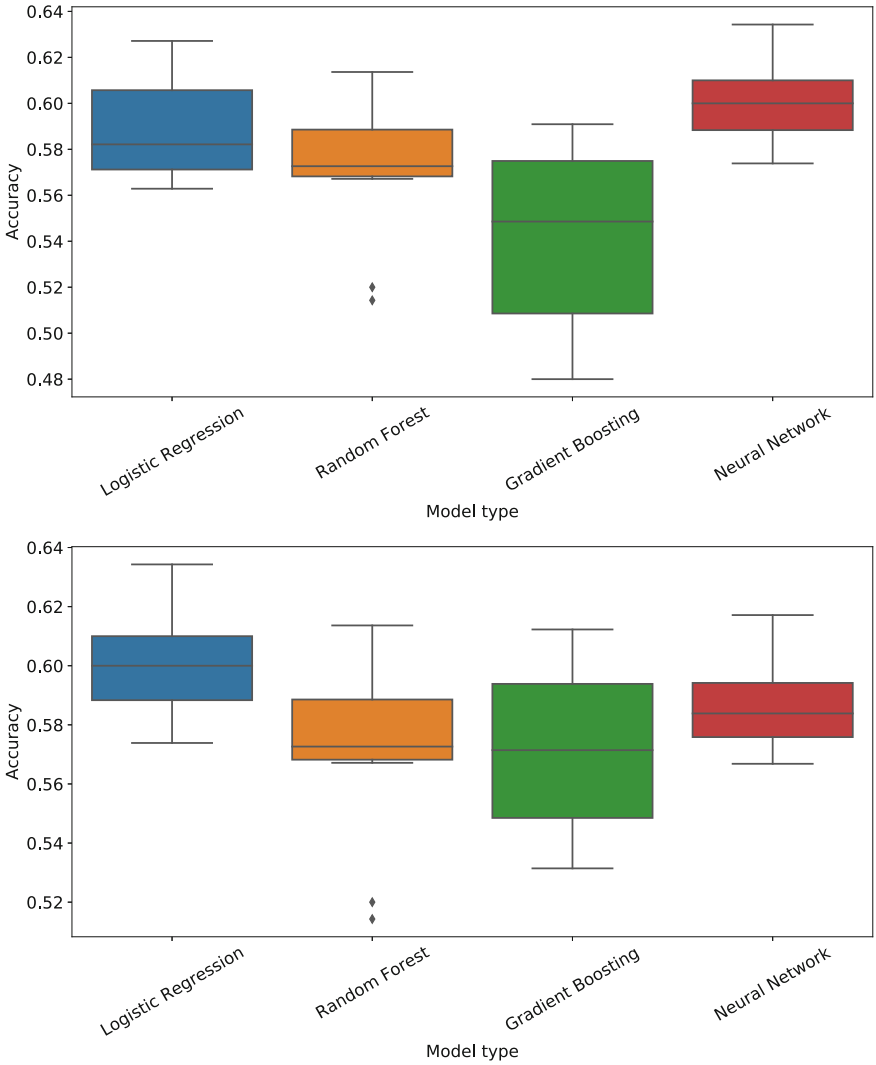


Fig. 11 A comparison of accuracy for different machine learning models: transferred models comparison (top) and ensemble model comparison (bottom)

5 Summary and Future Work

In this chapter, we suggested the Ensembled Transferred Embeddings learning framework. The proposed framework is composed of four main steps: (1) Manually label a small sample dataset (2) Extract embeddings from related large-scale labeled datasets (3) Train transferred models using the extracted transferred embeddings

and the labels of the sample dataset (4) Build an ensemble to combine the outputs of the different transferred models into a single prediction We then showed the applicability of the proposed framework for the item categorization task in settings for which the textual attributes representing items are noisy and short, and labels are not available.

We evaluated our method using a large-scale real-world invoice dataset provided to us by PayPal. The results of this evaluation showed that our method significantly outperforms other traditional (e.g., TF-IDF) as well as state-of-the-art (e.g., methods based on general purpose pretrained models such as BERT) item categorization methods. We believe that the reason for the superiority of our method is due to the unique characteristics of the text used in our setting—item descriptions are short, noisy and are domain specific. These characteristics make the text distribution in our setting different than that of the text used to train general purpose embeddings. Consequently, this makes our embeddings more relevant for the task at hand, despite being trained with a much smaller dataset.

One limitation of our method is the need to manually label a high quality sample dataset. As we explained in the chapter, generating a high quality dataset, even when using a crowdsourcing service, is not a trivial task. A non-negligible effort should be invested in properly designing the crowdsourced labeling task, in order to obtain an accurate result, in a relatively fast and cheap process.

Another limitation of our method is the use of domain knowledge. In our framework domain knowledge is essential for the selection of the related large-scale labeled datasets. As demonstrated throughout the chapter, choosing datasets that are related to the domain and the task at hand is essential to building more accurate models.

In future work it would be interesting to investigate the following directions:

- **Additional domains and tasks.** We believe that the proposed ETE framework is generic enough and can be very useful in additional text classification domains such as health, law, social media, etc. We also believe that the ETE framework, with some adjustments, can be beneficial to tasks other than text classification, such as computer vision, audio processing, time series analysis, and more.
- **Combining more methods for text classification with small data.** In recent years methods such as active learning and semi supervised learning showed great success in reducing the number of labeled samples needed for an accurate text classification. These methods can be integrated into the ETE framework in order to achieve better accuracy.
- **Uninformative items detection.** 15% of the items that were annotated using Mechanical Turk were labeled as uninformative. A production item categorization system would need to filter out such items as a first step. An interesting research direction would be to build a machine learning model to identify such uninformative items, perhaps using the same ETE framework.

- **Using additional item attributes.** In this work we used only the textual attributes: item name and item description, as input for our model. Using other attributes such as price, seller ID, and user ID was shown in (Krishnan & Amarthaluri, 2019) to improve the accuracy of the item categorization model.

Acknowledgements This research was funded by PayPal. We would like to thank our colleagues from PayPal: Yaeli, Adam, Omer, and Avihay who provided meaningful insights and greatly assisted in improving this work.

References

Baldwin, T., de Marneffe, M.-C., Han, B., Kim, Y.-B., Ritter, A., & Xu, W. (2015). Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the workshop on noisy user-generated text* (pp. 126–135).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, *11* (Feb), 625–660.

Hadar, Y., & Shmueli, E. (2021a). Categorizing items with short and noisy descriptions using ensembled transferred embeddings. *Expert Systems with Applications*.

Hadar, Y., & Shmueli, E. (2021b). *Source code for ensembled transferred enbeddings.* https://github.com/h-yonatan/Ensembled-Transferred-Enbeddings. (Accessed: 2021-05-27)

Hedderich, M. A., Lange, L., Adel, H., Strötgen, J., & Klakow, D. (2020). A survey on recent approaches for natural language processing in low-resource scenarios. *arXiv preprint arXiv:2010.12309*.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., et al. (2015). Skip-thought vectors. In *Advances in neural information processing systems* (pp. 3294–3302).

Kozareva, Z. (2015). Everyone likes shopping! multi-class product categorization for e-commerce. In *Proceedings of the 2015 conference of the north American chapter of the association for computational linguistics: Human language technologies* (pp. 1329–1333).

Krishnan, A., & Amarthaluri, A. (2019). Large scale product categorization using structured and unstructured attributes. *arXiv preprint arXiv:1903.04254*.

Li, M. Y., Kok, S., & Tan, L. (2018). Don't classify, translate: Multi-level e-commerce product categorization via machine translation. *arXiv preprint arXiv:1812.05774*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Ruder, S. (2019). *Neural transfer learning for natural language processing* (Unpublished doctoral dissertation). NUI Galway.

Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 806–813).

Werbin-Ofir, H., Dery, L., & Shmueli, E. (2019). Beyond majority: Label ranking ensembles based on voting rules. *Expert Systems with Applications*, *136*, 50–61.

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, *5* (2), 241–259.

Data Mining in Medicine

Beatrice Amico, Carlo Combi, and Yuval Shahar

1 Introduction

Extensive amounts of knowledge and data stored in medical databases require the development of specialized tools for data access, data analysis, knowledge discovery, and effective use of the discovered knowledge. The need to understand large and complex data sets has now increased in different fields. The traditional manual data analysis has become insufficient, and methods for efficient computer-based analysis indispensable. With these large amounts of data, the ability to extract useful knowledge hidden in these large amounts of data and perform actions on the basis of the discovered knowledge, is becoming increasingly important.

Knowledge discovery in databases is frequently defined as a process [1] consisting of the following steps: understanding the domain, forming the data set and cleaning the data, extracting regularities hidden in the data, thus formulating knowledge seen as patterns or models (this step is referred to as Data Mining), postprocessing of discovered knowledge, and exploiting the results.

Data mining in medicine has been receiving considerable attention since it provides a way of revealing useful information hidden in the clinical data. Nowadays technology gives the possibility to clinicians to collect automatically huge amounts of data. The analysis of such healthcare/medical data collections could greatly help to gain a deeper insight into the health conditions of the population and extract

B. Amico · C. Combi (✉)
Department of Computer Science, University of Verona, Verona, Italy
e-mail: beatrice.amico@univr.it; carlo.combi@univr.it

Y. Shahar
Department of Information Systems Engineering, Ben-Gurion University of the Negev, Beersheba, Israel
e-mail: yshahar@bgu.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_27

useful information that can be exploited in the assessment of healthcare/medical processes.

Within the context of the healthcare industry, the massive amount of data derived from electronic medical records, which are being under-utilized. Patient records collected data for diagnosis and prognosis and typically include values of demographic, clinical and laboratory parameters. Such data sets can be characterized by their incompleteness because of missing parameter values, incorrectness as a consequence of systematic or random noise in the data, sparseness as a result of few and/or non-representable patient records available, and inexactness in the selection of parameters for the given task [2]. An important aspect that has to be consider is the intrinsic necessity to take into account the temporal component. This aspect needs to be considered when representing information within computer-based systems, when querying information about temporal features of the represented real world, when reasoning about time-oriented data, during the design process of analysis tools for prediction, personalized medicine, and therapy support.

In general, data mining can be used for solving descriptive and predictive data mining tasks. Descriptive Data Mining tasks concern about finding human-interpretable patterns and associations, after considering the data as a whole and constructing, where typical methods include association rule learning, and (hierarchical or k-means) clustering. In contrast, Predictive Data Mining investigate to prefigure some response of interest. Starting from the entire data set, it aims at inducing a predictive model that holds on the data and can be used for prediction or classification of yet unseen instances [3, 4].

In summary, complex domains like the medical one may be prone to show hidden relationships inside the data. Data Mining techniques help to unravel and discover such hidden patterns and regularities for giving insights to users.

In this chapter, we consider different Data Mining techniques used for extracting knowledge from medical data. In Sect. 2 we will introduce the concept of Machine Learning, focusing on the application of Deep learning methods in different medical fields in order to improve quality of care, safety, diagnosis, and prognosis of patients. In Sect. 3 we will enhance the concept of temporal data mining with particular attention to discover temporal patterns, unexpected trends, or other hidden relations in a huge and overwhelming quantities of data. We decided to analyze a slice of the possible approaches for mining biomedical data. Instead, we overlooked the social data mining related, for example, to drug information extracted from social media and pharmaceutical databases. We also left out the visual data mining on temporal data for the exploration of data.

2 Machine Learning and Some Emerging Medical Applications

As we mentioned before, we are in the era of big data. This massive quantity of data calls for automated methods of data analysis, which machine learning provides [5].

Artificial intelligence can potentially provide improvements in many sectors, becoming a major element even in the healthcare and medical landscapes. Recent examples such as skin lesions, diabetic retinopathy, and radiology detection have highlighted the potential use of AI in medicine to improve quality, safety, and diagnosis.

Data Mining in medicine is most often used for building classification models, these being used for the typical medical decision-making tasks as diagnosis, prognosis, or treatment planning. Data mining techniques are able to perform predictive modeling, exploit the knowledge available in the clinical domain and propose innovations to support clinical decisions. Indeed, one of the goals of predictive data mining in clinical medicine is to derive decision models that can use patient-specific information which—once evaluated and verified—may be embedded within clinical information systems, to predict the outcome of interest and to there by supporting clinical decision-making [4].

The data mining baseline is grounded by research areas such as machine learning and deep learning.

Machine Learning (ML) consists of a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision-making under uncertainty [6].

Deep learning (DL) is a specialized branch of ML, corresponding to a group of different techniques for training in neural networks that deploy multiple hidden layers to solve the pattern recognition problems [7]. It is a branch of machine learning that attempts to model higher-level abstractions in data by using model architectures with non-linear transformations. Deep learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple modules into a representation at a higher, slightly more abstract level. The key aspect of deep learning is that these layers of features are not designed by humans but learned from data using a general-purpose learning procedure [8]. It is in the intersections among the research areas of neural network, graphical modeling, optimization, pattern recognition, and signal processing [9]. Deep learning allows computational models that are composed of multiple processing layers based on neural networks to learn representations of data with multiple levels of abstraction. Some of the most successful deep learning methods involve artificial neural networks, such as Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Deep Belief Networks (DBN), and Stacked Autoencoder (SAE). The major differences between deep learning and traditional artificial neural networks (ANNs) are the number of hidden layers, their connections, and the capability to learn meaningful abstractions of the inputs. In fact, traditional ANNs are usually limited to three layers and are trained to obtain supervised representations that are optimized only for the specific task and are usually not generalizable. Differently, every layer of a deep learning system produces a representation of the observed patterns based on the data it receives as inputs from the layer below, by optimizing a local unsupervised criterion. The key aspect of deep learning is that these layers of features are not designed by human engineers, but they are learned from data using a general-purpose learning procedure [10].

Deep learning offers the possibility to exploit different techniques that uncover the hidden opportunities and patterns in clinical data. There are many aspects of deep learning that could be helpful in healthcare/medicine, such as its superior performance, end-to-end learning scheme with integrated feature learning, capability of handling complex and multi-modality data, and so on. The deep learning research field must address several challenges relating to the characteristics of health care data; for this reason we have the necessity to implement methods and tools able to enhance the cooperation between health care information workflows and deep learning techniques. In this section, considering the scientific landscape in medical and healthcare applications, among different applications of deep learning in medicine, we focus on the medical imaging and the Electronic Medical Records (EHRs). We will discuss recent applications of deep learning in medicine, highlighting the key aspects to significantly impact in the healthcare domain.

2.1 Clinical Imaging

Deep learning has contributed a lot to the image analysis in the medical field. Deep learning techniques are used in the Medical Imaging field in four ways: classification, segmentation, regeneration, and detection [7].

In medical imaging, the accurate diagnosis and/or assessment of a disease depends on both image acquisition, which has improved significatively over the recent years devices acquiring data at faster rates and increased resolution, and image interpretation that has only recently begun to benefit from computer technology.

In particular, Convolutional Neural Networks (CNNs) have proven to be powerful tools for a broad range of computer vision tasks. Deep CNNs automatically learn mid-level and high-level abstractions obtained from raw data, in this case images. Medical image analysis is quickly entering the field of the CNNs and other deep learning methodologies through a wide variety of applications. Indeed, recent studies [11, 12, 13, 14] indicate that the generic descriptors extracted from CNNs are extremely effective in recognition and localization in clinical images. The main power of a CNN lies in its deep architecture, which allows for extracting a set of discriminating features at multiple levels of abstraction. Training a deep CNN from scratch is challenging. Firstly, CNNs require a large amount of labeled training data, a requirement that may be difficult to meet in the medical domain where experts' annotation is often approximated and the diseases are various. Secondly, training a deep CNN requires large computational and memory resources, without which the training process would be extremely time-consuming. Finally, training a deep CNN is often complicated by overfitting and convergence issues, which often require repetitive adjustments in the architecture or learning parameters of the network to ensure that all layers are learning with comparable speed.

Most of the medical images interpretations are performed by physicians. However, this kind of knowledge depends on levels of human subjectivity, as a consequence of variation across different interpretations. Many diagnostic tasks

require an initial search process to detect abnormalities, quantify measurements and changes over time. The benefits of automated tools are the potential to improve diagnosis, by facilitating identification of the findings that require treatment and to support the medical decision [15].

Observing the panorama of the medical applications of deep learning techniques in medical images, we spotlight on three main diseases: diabetes retinopathy, breast cancer and lymph node metastasis, and skin cancer. On the other hand, considering data from EHR, we focus our attention on readmissions, hospitalizations, and prognosis.

Considering this scenario, a spontaneous question could arise: can deep networks be used effectively for medical tasks? Can we rely on learned features alone or may we have to combine them with additional knowledge to reach the goal?

2.1.1 Diabetic Retinopathy (DR)

Diabetes mellitus is a group of metabolic diseases characterized by chronic hyperglycemia resulting from defects in insulin secretion, insulin action, or both [16]. Diabetes leads to a range of complications grouped into macrovascular (large blood vessel) complications, such as cardiovascular disease and stroke, and microvascular (small blood vessel) complications, such as kidney disease. People with diabetes fear visual loss and blindness more than any other complication. Diabetic retinopathy (DR) is a specific microvascular common complication of diabetes, is the major cause of vision loss among middle-aged men and women, and elderly people in many countries [17].

Gulshan et al. [18] established a clear path toward the use of AI, not to replace physicians, but rather to perform simple, cost-effective, and widely available examinations and analyses that could help identify at-risk patients who require referral for specialty care, and reassuring other patients that potential retinal manifestations of their diabetes are not present or are stable. The authors implemented a deep CNN trained using more than 128,000 retinal fundus images from adult patients with diabetes to identify referable diabetic retinopathy. The network used a function that first combines nearby pixels into local features, then aggregates those into global features. All images in the development and clinical validation sets were graded 3 to 7 times for diabetic retinopathy, diabetic macular edema, and image gradability by a panel of 54 US licensed ophthalmologists and ophthalmology senior residents. The algorithm was validated using two data sets, EyePACS-1, a data set consisted of 9963 images from 4997 patients obtained in the United States (prevalence of RDR: 7.8%) and Messidor-2, a data set had 1748 images from 874 patients obtained at 3 hospitals in France (prevalence of RDR:14.6%), both graded by at least 7 US board-certified ophthalmologists with high intra-grader consistency. It computed diabetic retinopathy severity from the intensities of the pixels in a fundus image. They defined sensitivity and specificity of the algorithm for detecting referable diabetic retinopathy (RDR), generating them based on the reference standard of the majority decision of the ophthalmologist panel. The algorithm was evaluated at two

operating points selected from the development set, one selected for high specificity and another for high sensitivity. Creating or "training" this function required a large set of images for which the diabetic retinopathy severity was already known (training set). During the training process, the parameters of the neural network were initially set to random values. Then, for each image, the severity grade given by the function was compared with the known grade from the training set, and parameters of the function were then modified slightly to decrease the error on that image. This process was repeated for every image in the training set many times over, and the function "learned" how to accurately compute the diabetic retinopathy severity from the pixel intensities of the image for all images in the training set.

Abramoff et al. [19] reported a pivotal trial leaded the central focus on the detection in a general practitioner's office of more than moderate diabetic retinopathy (mtmDR). They enrolled 900 subjects from 10 primary care practice sites throughout the USA. They were asymptomatic diabetes patients, with an age range of 22 to 84 years, not previously diagnosed with DR. The autonomous AI system, IDx-DR, had two core algorithms, an Image Quality AI-based algorithm, and the Diagnostic Algorithm. The image quality algorithm was implemented as multiple independent detectors, a multilayer convolutional neural networks, for retinal area validation as well as focus, color balance and exposure, and used interactively by the operator to detect, in seconds, sufficient image quality for the Diagnostic algorithm to rule out (or in) more than mild diabetic retinopathy (mtmDR), and thus maximized the number of subjects that can be imaged successfully. The diagnostic algorithm was a clinically inspired algorithm, incorporating independent, validated detectors for the lesions characteristic for DR, including microaneurysms, hemorrhages, and lipoprotein exudates. The images were classified according to the Early Treatment Diabetic Retinopathy Study (ETDRS) Severity Scale. They obtained an observed Sensitivity to fundus mtmDR of 87.4% and a Specificity of 89.5%. Based on the achieved results, for the first time FDA authorized an AI diagnostic system in any field of medicine, with the potential to help prevent vision loss in thousands of people with diabetes annually.

2.1.2 Breast Cancer and Lymph node Metastasis

Breast cancer is the most frequent malignancy in women worldwide. A positive screening mammography, or the development of breast symptoms or breast changes, requires appropriate diagnostic evaluation [20].

In the context of the researcher challenge competition CAMELYON16 [21], the authors described a set of algorithms for the diagnostic assessment to detect the lymph node metastasis in women breast cancer. The CAMELYON16 challenge highlighted a significant opportunity for AI in pathology, namely assisting pathologists with screening for lesions in histopathologic sections. They applied two approaches to compare the performances of the algorithms generated by automated deep learning systems for the detection of nodal metastases in the whole-

side images. The first method involved a panel of 11 pathologists with varying degrees of expertise in breast pathology, who were given 2 h to review all 129 test slides, less than 1 min per slide. The second method was to ask one pathologist to review all cases without a limit of time. Confidence in the algorithms came from their ability to detect metastases. The top algorithms performed better than the 11 pathologists with time constraints at identifying micro metastases but were not statistically different when compared with the performance of the pathologist with unlimited time. A training data set of whole-slide images from 2 centers in the Netherlands with ($n = 110$) and without ($n = 160$) nodal metastases verified by immunohistochemical staining were provided to challenge participants to build algorithms. The algorithms performance was evaluated through an independent test set of 129 whole-slide images (49 with and 80 without metastases). The CAMELYON16 challenge demonstrated that some deep learning algorithms were able to achieve a better AUC then the panel of 11 pathologists who had time constraints, demonstrating that interpretation of pathology images can be performed by deep learning algorithms at an accuracy level that rivals human performances.

Another application of deep learning techniques in breast cancer is described in [22]. One of the main early signs on mammograms is the appearance of microcalcifications, whose diameter ranges from 0.1 to 1 mm. Early detection and accurate identification of malignant microcalcifications can facilitate early detection, diagnosis, and timely treatment of breast cancer. Due to the small size and low contrast compared to the background of images, it is difficult for radiologists to make objective and accurate evaluation of microcalcifications. Consequently, there is a need to develop helpful automated tools to overcome these problems and improve diagnostic performances. Microcalcifications are highly correlated with breast cancer, therefore, the aim of this investigation was to evaluate the performance of an innovative deep learning model for classifying breast lesions. This study concerned reviewed mammograms from 1204 female patients histopathologically diagnosed with benign or malignant breast lesions at the SunYat-sen University Cancer Center (Guangzhou, China) and Nanhai Affiliated Hospital of Southern Medical University (Foshan, China). In this work, we see the application of another deep learning algorithm, named SAE, i.e., a stacked autoencoder (SAE) creates a deep network by stacking multiple autoencoders hierarchically. Each autoencoder is a specific type of feedforward neural networks. They compress the input into a lower-dimensional code and then reconstruct the output from this representation; the output of each autoencoder is used as the training set for the next autoencoder. More specifically, in an SAE within layers, the first layer is trained as an autoencoder to obtain the first hidden layer, and the output of the k hidden layer is used as the input of the $(k + 1)$ hidden layer. The training group consisted of 1000 images, including 677 benign and 323 malignant lesions. The test group consisted of 204 images, including 97 benign and 107 malignant lesions. Data about microcalcifications and suspicious breast masses were extracted through a segmentation pipeline. Proposed method demonstrated to be effective, and it could accurately detect and extract

suspicious microcalcifications from the background of a low-density image. The results demonstrated that deep learning not only enabled accurate segmentation of microcalcifications but also provided an efficient analysis of their characteristics, leading to a marked improvement in discriminating between benign and malignant breast lesions.

2.1.3 Skin Cancer

Skin cancer, the most common human malignancy, is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis, a biopsy, and a histopathological examination.

Esteva et al. [23] compared the ability of a deep convolutional neural network (CNN) to discriminate the most common skin cancers. They illustrated the equivalence in the performance of their algorithm against at least 21 dermatologists in evaluating biopsy-proven clinical images. Automated classification of skin lesions using images is a challenging task owing to the variability in the appearance of skin lesions. The authors proposed that mobile devices, such as smartphones, could be deployed with similar algorithms, permitting potentially low-cost universal access to vital diagnostic care anywhere in the world. They performed a classification of skin lesions using a single CNN based on a GoogleNet Inception v3 CNN architecture, trained end-to-end by images directly, using only pixels and disease labels as inputs. They trained the CNN using a data set of 129,450 clinical images, which were organized in a tree-structured taxonomy of 2032 diseases. Data were split into 127,463 training and validation images, and 1942 biopsy-labeled test images. They observed that most dermatologist performances were below the system ROC curves. So this algorithm can be seen as an example of fast, reliable, scalable diagnostic modality, which can be implemented on mobile devices.

The study reported in [24] represents a second example of the investigation of skin cancer with deep learning techniques. Melanoma, caused by abnormal reproduction of melanocyte cells, is the lethal form of skin cancer. Melanocytes cells are responsible for producing melanin pigments that give color to skin. Jafari et al. [24] proposed a deep CNN architecture for the extraction of lesion regions from skin images. The input images were generated by standard cameras; hence, they should be pre-processed in order to handle noise. They defined a local patch as a window around each pixel of the image, showing the nearby texture around the center pixel. In addition, they considered a zoomed-out window, with the same center as the local patch, for revealing the global structure of the region. The two patches of local and global texture fed the proposed CNN. The output of the CNN was a label for the central pixel of the patch. About 140,000 patches were obtained and used for training of the CNN. The used data set consisted of 126 digital images (66 melanoma, 60 non-melanoma).The number of patches that were selected from each training image is 1500, where half of them were randomly chosen from lesion region and the other half are randomly selected from non-lesion parts. Thus all these patches were extracted and used for training of the CNN in each run.

2.2 Electronic Medical Records

Deep learning techniques have been also applied to Electronic Health Records (EHR) data. Data-driven healthcare, defined as usage of those available big medical data to provide the best and most personalized care, is becoming to be one of the major trends to the success of revolutionize healthcare industry. One key aspect to the success of those medical applications is extracting effective features from patient EHRs, which is usually referred to as electronic phenotyping in medical informatics. At the beginning of this section we consider some examples on application of deep learning to EHRs. The global health care systems are rapidly adopting Electronic Health Records, which are systematic collections of longitudinal patient health information generated by one or more encounters in any care delivery setting. They represent a highly rich sources of patient data, which consists of both structured data such us information about medical history facts, medications, diagnosis, diagnostic exams, treatment plans, allergies, laboratory and test results, and unstructured data such as free-text clinical notes. Some proposals deal with applying deep learning to predict diseases from the patient clinical status as stored in the EHR. Most of the literature deals with processed EHRs of a health care system by a deep architecture for a specific, usually supervised, predictive clinical task.

Cheng et al. [25] proposed a deep learning algorithm for extracting meaningful features, or phenotypes, from patient EHRs. They first represented the EHRs for every patient as a temporal matrix with the time on one dimension and a kind of event on the other dimension. They used a four-layer CNN to extract phenotypes and perform prediction, exploited a framework composed by input layer, one-side convolution layer, max-pooling layer and softmax prediction layer. The first layer was composed of EHR matrices. The second layer was a one-side convolutional layer that can extract phenotypes from the first layer. The third layer was a max-pooling layer introducing sparsity on the detected phenotypes, so that only those significant phenotypes remain. The fourth layer was a fully connected softmax prediction layer. They validated the model using two specific clinical scenarios: Congestive Heart Failure (CHF) and Chronic Obstructive Pulmonary Disease (COPD).

Avati et al. [26] faced the problem of improving the quality of end-of-life care for hospitalized patients. They used a Deep Neural Network trained on the EHR data from previous years, to predict the mortality of patients within 3 to 12 months from that date, using EHR data of patients from the prior year as a proxy for patients that could benefit from palliative care. The criteria for deciding which patients benefit from palliative care can be hard to state explicitly. This approach used deep learning to screen patients admitted to the hospital, to identify those who were most likely to have palliative care needs. The algorithm addressed a proxy problem and used that prediction for making recommendations for palliative care referral.

From EHR data of 2 Million adult and pediatric patients from the Stanford Translational Research Integrated Database Environment (STRIDE) during 1995–2014, 221,284 patients correspond to the inclusion criteria. Specifically they used

a Deep Neural Network with an input layer of 13,654 dimensions, 18 hidden layers (each consisting of 512 dimensions) and a scalar output layer, a logistic loss function at the output layer and the Scaled Exponential Linear Unit (SeLU) activation function at each layer. They demonstrated that routinely collected EHR data could be used to create a system that prioritizes patients for follow up with palliative care. They created a model for all-cause mortality prediction and use that outcome as a proxy for the need of a palliative care consultation.

Rajkomar et al. [27] constructed a predictive statistical model using data from an electronic health record to drive personalized medicine and improve healthcare quality. Using a specific representation, the Fast Healthcare Interoperability Resources (FHIR), the authors demonstrated that Deep Learning methods are capable of accurately predicting multiple medical events from multiple centers without site-specific data harmonization. They used data from two US academic medical centers with 216,221 adult patients hospitalized for at least 24 h. The data included patient demographics, provider orders, diagnoses, procedures, medications, laboratory values, vital signs, and flowsheet data, which represent all the structured data elements (e.g., nursing flowsheets), from all inpatient and outpatient encounters. One data set additionally contained de-identified, free-text medical notes. A deep learning algorithm produced predictions for a wide range of clinical problems and outcomes and outperformed traditional, clinically used predictive models. Because the authors were interested in understanding whether deep learning could scale to produce valid predictions across divergent healthcare domains, they used three deep learning neural network model architectures, plus an ensemble method. Each model was trained on all four tasks, namely the prediction of a clinical outcome (death), a standard measure of quality of care (readmissions), a measure of resource utilization (length of stay), and a measure of understanding of a patient's problems (diagnoses), respectively.

The deep learning models achieved high accuracy for tasks such as predicting In-hospital mortality across the two sites (AUCs: 0.93, 0.95); 30-day unplanned readmission (AUCs: 0.75, 0.76); Prolonged length of stay (AUCs at 24 h: 0.85, 0.86); all of the patient's final discharge diagnoses (frequency-weighted AUC: 0.90). These results were better than the results of current traditional predictive models considered to be the state of the art for each task.

2.3 Challenges in Healthcare

Despite the promising results obtained using deep learning approaches, there remain several challenges related to the application of these algorithms to the healthcare domain. In particular, we highlight the following key issues [7, 10] :

- **Data Volume**: Deep learning is so successful in domains where huge amount of data can be easily collected. To produce satisfactory results, we need to have a considerable amount of data that feeds a comprehensive deep learning model.

Understanding diseases and their variability is much more complicated than other tasks, such as image or speech recognition.

- **Data Quality**: Unlike other domains where the data are clean and well-structured, health care data are highly heterogeneous, ambiguous, noisy, and incomplete. Training a good Deep Learning model with such massive and variegate data sets is challenging and needs to consider several issues, such as data sparsity, redundancy, and missing values. Specially when we work with EHR we have to consider [25] :
  - **High-Dimensionality.** There is a large amount of distinct medical events in patient EHRs. Those events also interact with each other.
  - **Temporality**. The patient EHRs evolve over time. The diseases are always progressing and changing over time in a nondeterministic way. The sequentiality of the medical events reveal important information on patient disease conditions. Designing Deep Learning approaches that can handle temporal healthcare data is an important aspect that will require the development of novel solutions.
  - **Varying sampling rate and granularity.** Time-oriented clinical data are often sampled at varying granularities—some might be measured each minute, such as heart rate or blood pressure in an Intensive Care Unit (ICU), while others might be measured once or twice a day, such as Temperature or White-Blood-Cell (WBC) Count; or even once in several weeks or months, as Hemoglobin A1C is measured in Diabetes, or even years, as height and Weight are measured in children.
  - **Synchronization.** Furthermore, different parallel, multivariate channels of information, such as different types of laboratory tests, imaging data, reported symptoms, and diagnostic codes, that accumulate in the patient's longitudinal record over time, need to be integrated and analyzed, while keeping the correct absolute time stamp or at least relative temporal distance among them. Handling such multivariate, varying-granularity longitudinal data is essential for effective analysis of clinical records.
  - **Sparsity**. The EHR data are extremely sparse. The data are largely missing in several ways, such as recording mistake, patient relocation, lack of visits, and so on.
  - **Irregularity**. Due to the complexity of patient diseases, there exists high variabilities among the EHRs of different patients, even with the same disease.
  - **Bias**. The above challenges, including systematic errors, can result in significant bias when health record data are used naively for clinical research. Moreover, it is important to note that an implicit population selection affects the data from which the output machine learning models are learned.
- **Domain Complexity**: Unlike other application domains (e.g., image and speech analysis), the problems in biomedicine and health care are more complicated.

The diseases are highly heterogeneous and for most of the diseases there is still no complete knowledge on their causes and how they progress.
- **Context-sensitive models**: the model might be specific to a particular medical problem or environment.
- **Interpretability**: Although Deep Learning models have been successful in quite a few application domains, they are often treated as black boxes, while in medicine is fundamental to explain why and how the algorithm works. It is a crucial step to convince clinicians about the actions recommended from the predictive system. In medicine the model performance and interpretability are equally important for health. Clinicians are unlikely to adopt a system they cannot understand.

Finally, moving to another challenge, the major issue about the introduction of AI in clinical practice could be identified in education. To facilitate this process, AI and other computational methods must be integrated into training programs, during which clinicians have the possibility to become comfortable with the combination of digital images and other data in with computer algorithms in their daily practice [28].

3 Temporal Data Mining in Medicine: Some Relevant Techniques

Modern technologies enable us to store huge and overwhelming quantities of data in the medical/healthcare domain. One of the main challenge in the analysis of Electronic Health Records is related to the sparsity and irregular sampling nature of medical data, since clinical data are commonly recorded only when patients enter the healthcare system, providing a rather sparse and biased view of the patient's clinical history. Extracting useful knowledge from these data is still a challenging task.

Disease and patient care processes often create characteristic states, trends, and temporal patterns in clinical events and observations. Identifying patient populations who share similar abstractions may be useful for clinical research, outcomes studies, and quality assurance [29].

To this end, Data Mining considers the temporal evolution of clinical data, biomedical time series, the change of medical information over time, and tries to extract useful temporal knowledge [30]. Analyzing data for supporting diagnostic tasks in medicine requires an explicit representation and consideration of the temporal semantics of data. The increasing use and availability of longitudinal electronic data presents a significant opportunity to discover new knowledge from multivariate, time-oriented data, by using various data mining methods.

In medicine, the analysis of time-oriented data enables researchers to discover new temporal knowledge and gain insight into the temporal behavior and temporal associations of such data, with the further objectives of clustering, classification,

and prediction [31]. An example from the medical domain would be supporting a clinical researcher who analyzes the results of a clinical trial of a new drug within a population of chronic patients.

As we mentioned in the previous section, there are two kinds of data mining models: predictive models and descriptive models. In this section we will focus on the descriptive modeling paradigm in healthcare. We will focus on Temporal Data Mining (TDM) approaches that were applied to the medical field, due to the intrinsic longitudinal nature of clinical data. In the following, we will introduce the basic concepts of temporal abstraction in medicine, which are the base for temporal association rules and of temporal pattern mining, we will then discuss. Finally, we will introduce the discovery of approximate temporal functional dependencies on clinical data.

Temporal data mining is a sub-field of data mining, in which various techniques are applied to time-oriented data to discover temporal knowledge about relationships among different raw data and abstract concepts, in which the temporal dimension is treated explicitly. Temporal data mining offers the possibility of obtaining a considerable understanding of various scientific phenomena and the potential for creation of accurate classification models [32].

The biomedical domain naturally has to deal with temporal issues due to the intrinsic longitudinal nature of clinical data. Further and innovative contributions are still being suggested to mine electronic health records for revealing novel patient-stratification principles or unknown disease correlations, to find dependencies in clinical databases, to retrieve suitable time parameterization for association and clustering experiments and to classify multivariate time series. Medical knowledge-based systems involve the application of medical knowledge to patient-specific data with the goal of reaching diagnoses or prognoses, deciding the best therapy regime for a patient, or monitoring the effectiveness of some ongoing therapy and if necessary, applying rectification actions. On the one hand, medical knowledge, like any kind of knowledge, is expressed in as general as possible, for example, in terms of associations, causal models of pathophysiological states, evolution models of disease processes, patient management protocols and guidelines. On the other hand, data on a patient include numeric measurements of various parameters (such as blood pressure, body temperature, etc.) at different time points. The difficulty is that often the abstraction gap between the specific, raw patient data, and the abstract medical knowledge does no longer allow any direct unification between information and knowledge. The process of data abstraction (see Fig. 1) aims to close this gap, bringing the raw patient data to the level of medical knowledge in order to permit diagnosis, prognosis, or therapeutic conclusions. So the data abstraction, in the context of medical problem solving, can be seen as an intelligent interpretation of the raw data that converges to new discovered knowledge [33].

The integration of the available data sources, properly pre-processed, leads to a uniform representation of the data as temporal sequences of healthcare events. There are different types of abstractions. In this chapter we give particular attention to a framework for temporal abstractions and to the related mining of patterns, association rules and functional dependencies.

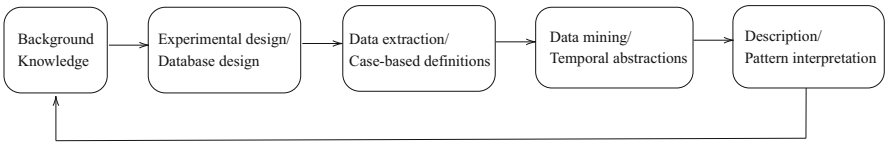


Fig. 1 Data abstraction process: from raw data to abstract knowledge

In clinical domains, what is often needed is a coherent intermediate level interpretation of the relationships between data and events. Often temporal data include a mixture of time-stamped raw data, time points, time intervals that could be either a part of the original raw input data or abstractions derived from them.

The goal is to abstract the data into higher-level concepts useful for one or more tasks, for example, the planning of a therapy or the summarization of a patient record. Most stored data include a time stamp in which the particular item is valid; an emerging pattern over a stretch of time has much more significance than an isolated finding or even a set of findings. A meaningful comprehension cannot use only time points, such as dates when data were collected; it must be able to characterize significant features over periods of time. Temporal abstraction enables us to overcome much of the problems of varying-frequency measurements and recordings, and of minor measurement errors and deviations in the raw data, through the creation of concepts (symbols) that are no longer time-stamped, raw data, but rather interval-based abstractions, or interpretations, of these data, and through the smoothing effect of these abstractions. Figure 2 helps understand the classification problem for various time-point series and time-interval series, representing various temporal variables within the same input data set. Different cases are listed below:

- Samples acquired at fixed frequency, often used for automated sampling (case a, b)
- Samples acquired at random frequency, often occurring for manual measurements (case c)
- Time intervals where the duration of the events is constant (ex: medication-administration periods) (case d)
- Time intervals where the temporal duration is varying (case e)

In many instances, the temporal abstraction process also alleviates the problem of missing values, through a process of interpolation across temporal gaps that is inherent in several of the temporal abstraction methods [35].

3.1 Temporal Abstractions and the Knowledge-Based Temporal Abstraction (KBTA) Method

Temporal abstractions (TAs) represent a crucial aspect in the context of time-oriented clinical monitoring, therapy planning, and exploration of clinical databases.

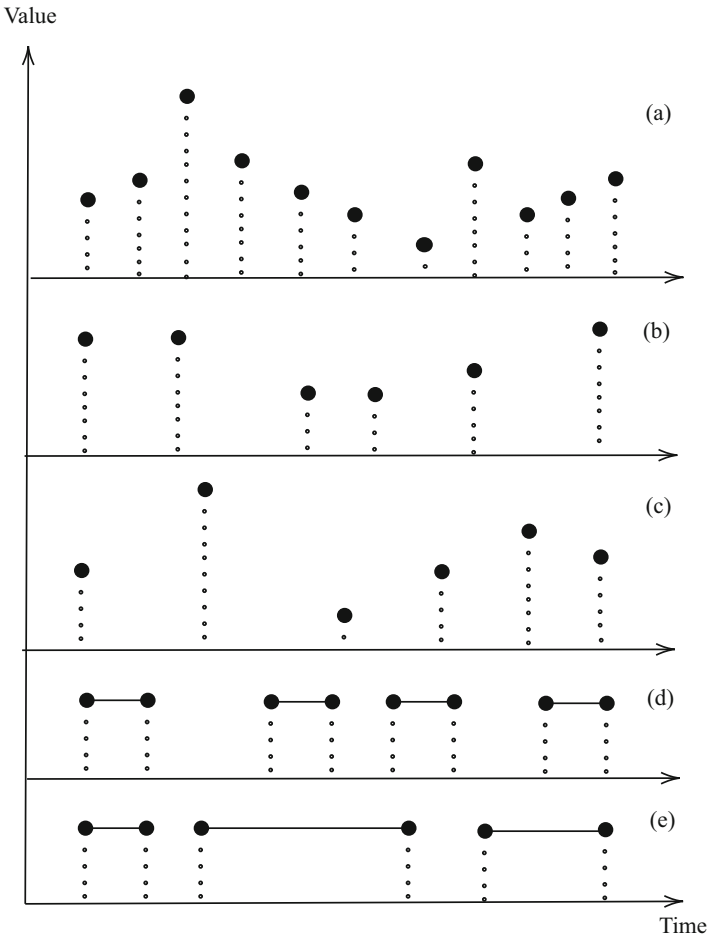


Fig. 2 The multiple formats of input data for time series analysis: time-point (**a**, **b**, **c**) and time-interval data (**d**, **e**) (adapted from [34])

They are able to give common representation of heterogeneous data, transforming raw clinical time series into meaningful representation of clinical events. In general, TAs represent a way to perform a shift from a quantitative time-stamped representation of raw data to a qualitative interval-based description of time series, with the main goal of abstracting higher-level concepts from time-stamped data. A TA provides a description of a set of timeseries through sequences of temporal intervals that correspond to relevant patterns that are detected in their time courses. Going in details, we can define a *Temporal abstraction (TA)* as a segmentation and/or aggregation of a series of raw, time-stamped, multivariate data into a symbolic time-interval series representation, often at a higher level of abstraction, suitable for human inspection or for data mining.

The temporal abstraction process can use knowledge-based approaches, which exploit domain-specific knowledge, or data-driven, domain-independent discretization methods. The patterns that can be formed by noting the temporal relationships among the symbolic time intervals are at least as interesting as the temporal abstractions that the symbolic intervals represent. These patterns essentially characterize frequent temporal pathways, trajectories, or journeys, within a given population of entities described by longitudinal multivariate data. Thus, in the case of the medical domain, they might be viewed as creating a temporal clustering of chronic patients who have a particular condition, according to the course of their disorder. The use of symbolic time intervals to abstract the raw, time-stamped data can reduce inherent random noise in the data (due to the discretization), avoid problems resulting from sampling the data at different frequencies and at various temporal granularities, and overcome missing data [32].

The first conceptual model based on TAs was proposed by Shahar in [36], with a method called the Knowledge-Based Temporal Abstraction (KBTA). This method exploits domain-specific knowledge, to generate state abstractions, as well as more complex abstractions, such as gradients. It has been applied in multiple domains, such as medicine, transportation, and cyber-security, for purposes such as interpretation, monitoring, visual exploration, classification, and prediction. The KBTA framework emphasizes the explicit representation of the knowledge required for abstractions of time-oriented clinical data and facilitates its acquisition, maintenance, reuse, and sharing.

Shahar and Musen [37] defined a general problem-solving method for interpreting data in time-oriented, knowledge-intensive domains, such as clinical ones: the knowledge-based temporal abstraction (KBTA) method. This method is able to acquire the relevant knowledge and to define the domain ontology (e.g., security ontology) based on five KBTA entities and the relations between them (primitive parameters, abstract parameters, contexts, events, and patterns). Five inference mechanisms are then applied in parallel for deriving the high-level abstractions from the raw data. The inputs to these five mechanisms are the primitive parameters and the events, which are related to raw data, and the outputs are contexts, abstractions, and patterns.

Shahar [36] defined the temporal abstraction (TA) task as follows: the input includes a set of time-stamped clinical parameters, relevant events and abstraction goals; the output includes a set of interval-based abstractions which should be relevant for clinical decision-making purposes in the given or implied clinical context, context-specific parameters at the same or at a higher level of abstraction and their respective values. The structure $\{< parameter, value, context >, interval\}$ denotes that the logical proposition "the *parameter* has a particular *value* given a specific interpretation *context*" is interpreted over a specific time *interval*. Such a structure is called an *abstraction*. The goal of the TA task is to evaluate and summarize the state of the patient over a period, to identify problems, to assist in a revision of an existing therapy plan, or to support the generation of a new plan. In addition, clinical guidelines (skeletal plans for therapy) can be represented as

TA patterns to be achieved, maintained, or avoided. Finally, generating meaningful abstractions supports explanation of a decision-support system's plans to its users.

TAs can be further mined to discover frequent (i.e., above some pre-defined level of support) temporal patterns, such as "(A before B) and (B overlaps C)," and in general, patterns using all of Allen's 13 interval-based temporal relations. A highly efficient method for fast mining of interval-based patterns, by exploiting the transitivity of temporal relations and a highly efficient indexing scheme, referred to as the KarmaLego algorithm, was proposed by Moskovitch and Shahar and demonstrated in multiple medical and non-medical domains as significantly faster than other, common alternatives [32].

Furthermore, the interval-based temporal patterns discovered by the KarmaLego algorithm, or by other methods for discovery of frequent interval-based temporal patterns, can be used as features for multiple machine learning algorithms. Indeed, Moskovitch and Shahar had demonstrated the effectiveness of this approach for a variety of classification and prediction tasks in several different clinical domains, using as features patterns discovered through the use of the KarmaLego algorithm, and detected within each instance to be classified, using a version of that algorithm, for single records, SingleKarmaLego [35]. Importantly, Shknevsky, Shahar, and Moskovitch had also demonstrated [31] that the same interval-based temporal patterns tend to be discovered in longitudinal patient records, within the same medical domain, across several different medical domains. Thus, using the patterns as classification and prediction features in various classifiers, is quite justified.

Moskovitch and Shahar had further extended the use of temporal abstractions and discovered interval-based temporal patterns, into the Temporal Discretization for Classification (TD4C) methodology [34]. The authors first present a method that learns in a supervised manner how to optimally discretize each temporal variable, in order to transform the time-point series into a time-interval series, mine these time-interval series, and perform classification.

The TD4C methodology essentially performs a grid-based search of possible discretization ranges on each time-oriented variable and uses them to perform a Temporal Abstraction of the State type, similar to that of Shahar's KBTA method [36], that has from two to five discrete values. The cut-off values whose distribution best differentiates between the target classes are chosen. The TD4C method is thus a supervised learning method, in which the discretization cutoffs are chosen so as to abstract the temporal data in a manner that creates the most differentiating distribution of the resulting states, amongst the entities that are classified by the various possible class values, for each of the time-oriented variables (concepts).

They then proceed to evaluate the method within several different clinical domains. They developed a framework for classification of multivariate time series analysis, which implements three phases: (1) application of a temporal abstraction process that transforms a series of raw time-stamped data points into a series of symbolic time intervals (based on either unsupervised or supervised temporal abstraction); (2) mining these time intervals to discover frequent temporal-interval relation patterns (TIRPs), examining the use of Allen's original seven temporal

relations, versus the use of an abstract version, including only three temporal relations; (3) using the patterns as features to induce a classifier.

The TD4C framework was evaluated on data sets in the domains of diabetes, intensive care, and infectious hepatitis. Using only three abstract temporal relations resulted in a better classification performance than using Allen's seven relations, especially when using three symbolic states per variable. Similarly when using the horizontal support (number of TIRP instances detected within the record to be classified) and mean temporal duration, as the TIRPs feature representation, rather than a binary (simple Yes/No existence) representation. The classification performance when using several different versions of TD4C was superior to the performance when using several standard unsupervised discretization methods [34].

Finally, building on the KBTA and KarmaLego studies, Sheetrit et al. [38] defined the Temporal Probabilistic proFiles [TPF] method for classification or prediction. The TPF method abstracts raw data concepts in each longitudinal record, discovers frequent interval-based temporal patterns, based on these abstractions, and creates, for each longitudinal record, a probabilistic distribution of the frequent patterns found in it. When a new longitudinal record needs to be classified, it is compared to either (i) the TPFs of all of the previously labeled single instances or (ii) to the aggregate TPF distribution of each of the target classes (e.g., the various clinical outcomes). The distance between the TPF distribution of the new instance to be classified and the TPFs of previously known instances (or, in the second case, to the aggregate TPFs of the various possible classes of instances) is then assessed by a version of the Kullback–Leibler divergence measure. In the first case, the top K instances that are most similar to the new instances are found and are used to determine the class predicted for it (e.g., using their majority), as is common on K-Nearest-Neighbor (KNN) methods. In the second case, the target class whose TPF distribution is most similar to that of the new instances is directly determined. Sheetrit et al. had demonstrated the effectiveness of the TPFs-based method for predicting sepsis in the intensive care unit (ICU), a key task in the ICU domain. The performance of the Top-K version of the TPF method was higher than the performance of both a known method exploiting frequent temporal patterns as features, and a deep learning method specialized for the ICU sepsis-prediction task, while the performance of the much more economic (in terms of computational resources) Class-based TPF version proved to be almost as good as the Top-K method, which uses all available labeled instances [38].

3.2 Association Rules

Healthcare organizations are increasingly collecting large amounts of data related to their day-by-day activities. The analysis of such healthcare databases could greatly help to gain a deeper insight into the health condition of the population and to extract useful information that can be exploited in the assessment of healthcare processes.

In clinical databases, the temporal features play the primary role [39]. Sometimes clinical data are represented by a set of time series of numeric values. In order to get a uniform representation of these data as temporal sequences of events, the clinical data needed first to undergo to a pre-processing procedure. Association rule mining is a method for identifying strong relations in a data set based on some measure of interestingness (e.g., confidence/precision, support or lift). Typically, such relations are expressed in terms of if-then rules consisting of different rule antecedents (conditions) and consequents (targets). It represents a technique with a lot of popularity in data mining research, including medical data mining. The strength of this technique is the possibility to explore completely all patterns that occur in the data. The disadvantage is that the number of association rules could be very large and the outputs could be difficult to deal with. Hence, it is desirable to reduce the mined rule set as much as possible while preserving the most important relations found in the data.

Temporal Association Rules (TARs) are association rules of the kind $A \rightarrow C$, where the antecedent (A) is related to the consequent (C) by some kind of temporal operator. TARs mining algorithms are aimed at extracting frequent associations, where frequency is evaluated on the basis of suitable indicators, the most utilized being support and confidence. The support gives an indication of the proportion of cases verifying a specific rule in the population; confidence instead represents the probability that a subject verifies the rule given that he verifies its antecedent.

Sacchi et al. [40] presented an approach to pre-process and interpret clinical time series. Their idea was to filter the original time series using temporal abstractions an then to interpret the new and derived time series by both statistical and artificial intelligent methods. Patterns of interest can be specified on the basis of domain knowledge into a set called Abstractions of Interest, and rules containing such pattern in the antecedent and in the consequent are extracted. After the development of a TARs mining framework mainly oriented to the analysis of clinical data, the framework had been extended to incorporate also administrative healthcare information into the data set. The authors, starting from the framework of knowledge-based Temporal Abstraction [36], proposed a new kind of temporal association rule working on the extraction of the frequent temporal precedence occurrences between patterns. Discovering occurrences of temporal relationships between patterns characterizing a time series needs the accomplishment of three conceptual and procedural steps. First of all, it is necessary to define the patterns and retrieve them in the time series; then a formal definition of the relationships of interest must be given and, finally, an algorithm to search for frequent occurrences of such relationships in the data set must be designed, implemented and run. Intuitively, a pattern is a behavior or property that we may want to distinguish in the data. In temporal data, a pattern is usually associated with a time interval in which such behavior occurs. Moreover, a pattern is often related to a qualitative representation of the property that we are looking for, which may be interesting in the problem domain. Here, according to the data model proposed in [41], temporal data are represented as time-stamped entities, called events, while their abstract representation is given by TAs as a sequence of intervals, called episodes.

Each episode corresponds to a specific behavior of interest detected in the time course of the data. TA tasks could be divided into two subtasks, each one solved by specific mechanisms: Basic TA, solved by mechanisms that abstract time-stamped data into intervals, and Complex TAs, solved by mechanisms that abstract intervals into other intervals. Complex TAs are used to detect patterns characterized by behaviors which cannot be represented by basic TAs. The episode set was evaluated according to a pattern specified between episodes of the two composing episode sets. The complex TA patterns were based on temporal relationships: more specifically, the temporal relationships investigated correspond to the 13 temporal operators defined in Allen's algebra [42]. They included: BEFORE, FINISHES, OVERLAPS, MEETS, STARTS, DURING, their corresponding inverse relations, and the EQUALS operator. In addition, as the two series of intervals used as input to a complex abstraction can originate both from the same and from different time series. We can thus exploit this kind of TA to detect a great variety of patterns. Figure 3 shows patterns of complex shape which have been detected both on a single time series, and on multiple time series.

Combi et al. [43] exploited knowledge-based Temporal Abstractions (TAs) to shift from a time-point quantitative representation of time series to a qualitative interval-based description of the available data.

A temporal fact f represents a class of episodes of the same type. Each episode e is associated with the interval when the episode holds. $e.start, e.end$ denote the starting and ending point of the interval associated with e, respectively. E_f denotes the set of episodes of a temporal fact f.

Informally we can enunciate: A *temporal association rule (TAR)* is a temporal pattern that exists between episodes of *temporal facts* belonging to a reference set FoI (Facts of Interest). To specify a TAR it is necessary to introduce the concept of temporal precedence: relation $precedes$ between two intervals a and b: $a \preceq b \iff a.start \leq b.start \wedge a.end \leq b.end$.

Definition 1 (Temporal Association Rule TAR) A TAR is an implication of the form $\{a_1, \ldots, a_n\} \xrightarrow{p} c$ where $\{a_1, \ldots, a_n\} \subset FoI, c \in FoI$ with $c \notin \{a_1, \ldots, a_n\}$, and $p = \langle LS, GAP, RS \rangle$ is the parameter set determining the relation between the antecedent and the consequent.

To determine an occurrence of the antecedent, there must exist a non-empty intersection between all the episodes e_i corresponding to facts a_i, respectively. More specifically, a (composite) antecedent occurrence has interval [*maxStart*, min*End*] where [40]:

$$max.Start \stackrel{\text{def}}{=\!=} \max(e_i.start | \, 1 \leq i \leq n), \; minEnd \stackrel{\text{def}}{=\!=} min(e_i.end | \, 1 \leq i \leq n).$$

Set p is composed by the following parameters: (i) *Left Shift (LS)*: maximum distance allowed between $maxStart$ and $c.start$; (ii) *Gap (GAP)*: maximum distance allowed between $minEnd$ and $c.start$; (iii) *Right Shift (RS)*: maximum distance allowed between $minEnd$ and $c.end$.

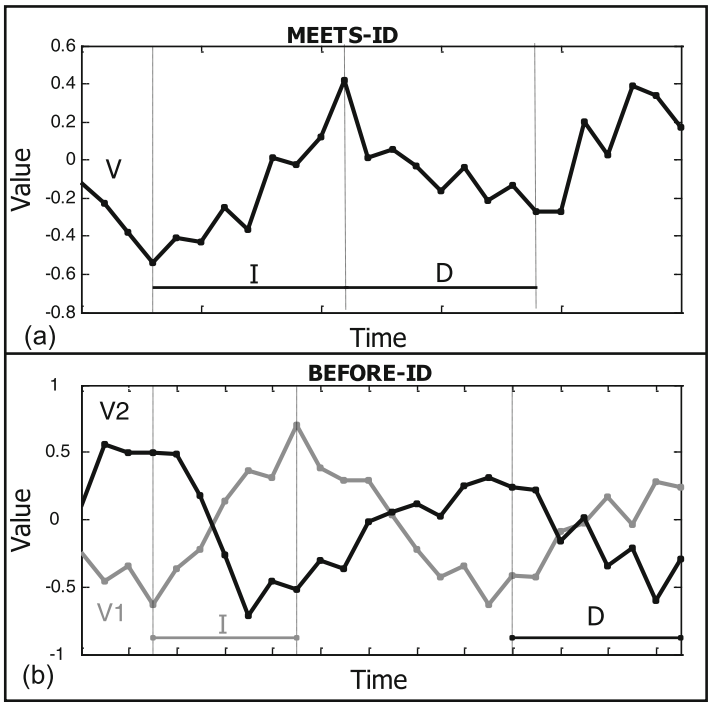


Fig. 3 Complex TAs used to detect patterns of complex shape both **(a)** on a single time series MEETS-ID, and **(b)** on multidimensional time series BEFORE-ID: in this case an I (increase) episode in V1 occurs before a D (decrease) episode in V2 (from [40])

Definition 2 (Occurrence of a TAR) An episode set $\{e_1, \ldots, e_n, e_c\}$ is an occurrence of a TAR $\{a_1, \ldots, a_n\} \xrightarrow{p} c$ with $p = \langle LS, GAP, RS \rangle$, if (i) $\{e_1, \ldots, e_n\}$ is an antecedent occurrence and e_c is a consequent episode; (ii) the antecedent occurrence precedes the consequent episode, i.e., $[maxStart, \min End] \preceq c$; (iii) all the quantitative constraints imposed by p are satisfied.

We found another application of the temporal association rules in medicine, where the authors used data from the Regional Healthcare Agency (ASL) repository of Pavia, which maintains a central data repository storing healthcare data about the population of Pavia area [39]. They specifically focus the analysis on a sample of patients suffering from Diabetes Mellitus. They performed the mining of Temporal Association Rules (TARs) over a set of temporal sequences of hybrid events, i.e., events characterized by heterogeneous temporal nature. Starting from the idea considering a pattern as the occurrence of one or more contemporary events, here a TAR is defined as a relationship specified through a temporal operator which holds between an antecedent, consisting in a pattern of single or multiple cardinality, and a consequent, consisting in a pattern with single cardinality.

3.3 Temporal Pattern Mining

Temporal pattern mining is another way to derive new knowledge from huge amount of data; temporal patterns are time intervals in which one or more time series assume a behavior of interest. A pattern can be seen as a behavior or a property in the data that we want to consider for a possible novelty knowledge in the domain of interest.

Mantovani et al. [44] proposed a new kind of temporal patterns called *Trend-Event Patterns*, namely TE-Ps, a family of temporal patterns focused on the interaction of trends and events. There are many different temporal abstractions, one of them is represented by trends. Trend abstraction aims at detecting relevant changes and change rates in the temporal evolution of a parameter. Trend abstraction entails merge and persistence abstraction, in order to derive the extents where no change is observed in the value of the considered parameter.

A *trend* is formed by consecutive values of a given measurement attribute that are stationary, increasing, or decreasing under the constraint that all the values of such trend stay within a defined range. A pattern is usually associated with a time interval where such behavior occurs. Discovering temporal relationships between patterns is also another problem in the temporal domain.

A TE-P is a pattern formed by an event E and two different trends for the same parameter: one before E and one after E, called $trend^{pre}$ and $trend^{post}$, respectively; for example: "The increasing trend of the body temperature of a patient before the administration of Paracetamol and the drug administered determine the decreasing trend of the body temperature of the same patient after such administration occurs" becomes $[Increasing; Paracetamol; Decreasing]$.

To derive a TE-P from this scenario, we need to start from the event, and more specifically from the time when such event happened. The parameter values are thus partitioned according to their *timestamp*. Every value before the event could potentially be part of $trend^{pre}$, while every values after the event could be in $trend^{post}$. In both trends the first parameter value is denoted as t_{start}, while the last one is t_{end}. The tuple in $trend^{pre}$ that is closer to the event is called t_{end}^{pre}, because it is the last tuple of it; while the tuple in $trend^{post}$ that is closer to the event is called $t_{star}t^{post}$, as it represents the beginning of such trend. Given the definition of TE-P: a TE-P is a pattern with an expression of the form: $[trend^{pre}; E; trend^{post}]$.

In a more formal way:

Given a trend tr, we denote with t_{start} the first tuple of tr and with t_{end} the last one. A trend tr is a non-empty set of parameter values $tr = \{t_{start}, \ldots, t_{end}\}$ satisfying $t_{start} \rightarrow \ldots \rightarrow t_{end}$. Implicitly, it means that $\forall t_i \in tr,\ t_{start}[VT] < t_i[VT] < t_{end}[VT]$.

Let t_{event} be the parameter value that denotes the event. Parameter values in $trend^{pre}$ are defined as follows: $\forall t_i \in tr^{pre}\ t_i[VT] < t_{event}[VT]$; while parameter values in $trend^{post}$ are defined as $\forall t_i \in tr^{post}\ t_i[VT] > t_{event}[VT]$.

Another example of temporal pattern mining is [45], where the focus was on the definition of methods and tools for the assessment of the clinical performance of hemodialysis (HD) service. In this paper the authors were interested in the

development of data mining tools for assessing the efficacy of the treatment delivered by a hospital hemodialysis department (HDD) on the basis of the process data routinely collected during hemodialysis sessions. Data were collected in order to assess the performance of the overall HDD, check the performance achieved for each patient and highlight problems and understand their reasons. This study aimed to discover the reason of the dialysis failures, deriving associations between monitoring variables and failures that may be interpreted as causal relationships. They defined a temporal data mining strategy consists of two steps: the extraction of basic temporal patterns, e.g., trends, from the median time series, and search for patient-specific associations between the temporal patterns and the failures. The obtained results demonstrated that the use of this kind of method in the context of an auditing system for dialysis management helped clinicians to improve their understanding of the patient behaviors.

3.4 Approximate Temporal Functional Dependencies

Knowledge in clinical databases may be derived by discovering patterns or regularities in data. By considering data stored in a plain relational database, we have another option to consider patterns or regularities: the functional dependencies (FDs).

Definition 3 (Functional Dependency FD) Let r be a relationship over the relational schema R: let $X, Y \subseteq R$ be sets of attributes of R. We assert that r fulfills the functional dependency $X \rightarrow Y$ (written as $r \models X \rightarrow Y$) if the following condition holds: $\forall t, t' \in r(t[X] = t'[X] \Rightarrow t[Y] = t'[Y])$.

Informally, for all the couples of tuples t and t' showing the same value(s) on X, the corresponding value(s) on Y for those tuples are (or must, if we are specifying a constraint) identical.

With FDs, we can express concepts such as *"for each patient with a given symptom the received treatment does not change"*:

$$Patient,\ Symptom \Rightarrow Treatment.$$

Considering the temporal features of the data, we pay our attention on some temporal extensions of FD.

3.4.1 Temporal Functional Dependencies (TFD)

Combi et al. [46] proposed a framework for TFD s that subsumes and extends the considered previous proposals. The proposed framework is based on a simple temporal relational data model based on the notion of temporal relation, i.e., a

relation extended with a timestamping temporal attribute VT , representing the valid time temporal dimension, i.e., the time when the fact is true in the real world [47].

Two temporal views have been introduced: they allow one to join tuples that represent relevant cases of (temporal) evolution. TFDs may be expressed by the syntax $[E - Exp(R), t - Group]X \rightarrow Y$ where $E - Exp(R)$ is a relational expression on R, called $evolution\ expression$, $t - Group$ is a mapping $\mathbb{N} \rightarrow 2^{\mathbb{N}}$, called $temporal grouping$, and $X \rightarrow Y$ is a functional dependency.
With TFDs, we can express concepts such as *"for each patient with a given symptom the received treatment does not change, considering a time windows of 10 days"*:

$$[10days]\ Patient,\ Symptom \Rightarrow Treatment.$$

A TFD is a statement about admissible temporal relations on a temporal relation schema R with attributes $U \cup \{VT\}$. A temporal relation r on the temporal relation schema R satisfies a TFD $[E - Exp(R), t - Group]X \rightarrow Y$ if it is not possible that the relation obtained from r by applying the expression $E - Exp(R)$ features two tuples t, t' such that (i) $t[X] = t'[X]$, (ii) $t[VT]$ nd $t'[VT]$ (and the valid times of their evolutions, if present) belong to the same temporal group, according to the mapping $t - Group$, and (iii) $t[Y] \neq t'[Y]$. In other words, FD $X \rightarrow Y$ must be satisfied by each relation obtained from the evolution relation by selecting those tuples whose valid times belong to the same temporal group.

Temporal grouping enable us to group tuples together over a set of temporal granules, based on one temporal dimension.

Four different classes of TFD have been identified [46]:

- *Pure temporally grouping* TFD: $E - Exp(R)$ returns the original temporal relation r. Dependencies of this class force the FD $X \rightarrow Y$, where $X, Y \subseteq U$, to hold over all the maximal sets which include all the tuples whose VT belongs to the same temporal grouping.
- *Pure temporally evolving* TFD: $E - Exp(R)$ collects all the tuples modeling the evolution of an object. No temporal grouping exists: that is, the temporal grouping collects all the tuples of r in one unique set.
- *Temporally mixed* TFD: the expression $E - Exp(R)$ collects all the tuples modeling the evolution of the object. The temporal grouping is applied to the set of tuples generated by $E - Exp(R)$.
- *Temporally hybrid* TFDs: First, the evolution expression $[E - Exp(R)$ selects those tuples of the given temporal relation that contribute to the modeling of the evolution of a real-world object (that is, it removes isolated tuples); then, temporal grouping is applied to the resulting set of tuples.

3.4.2 Approximate Functional Dependencies (AFD)

Considering FDs as a way of discovery properties on a specific set of data, we can extend the FDs to AFDs. Given a relation r where a FD holds for most of the tuples

in r, we may identify some tuples, for which that FD does not hold. Consequently, we define some measurements over the error we make in considering the FD to hold on r.

Kivinen et al. [48] defined three measurements:

- G_1: considers the number of violating couples of tuples. Formally:

$$G_1(X \rightarrow Y, r) = |\{(t, t') \ : \ t, t' \in r \wedge t[X] = t'[X] \wedge t[Y] \neq t'[Y]\}|.$$

  The related *scaled measurement* g_1 is defined as follows:

$$g_1(X \rightarrow Y, r) = G_1(X \rightarrow Y, r)/|r|^2,$$

  where $|r|$ is the cardinality of the relation r, i.e., the number of tuples belonging to the relation r.
- G_2: considers the number of tuples which violate the functional dependency. Formally:

$$G_2(X \rightarrow Y, r) = |\{t \ : \ t \in r \wedge \exists t'(t' \in r \wedge t[X] = t'[X] \wedge t[Y] \neq t'[Y])\}|.$$

  The related *scaled measurement* g_2 is defined as follows:

$$g_2(X \rightarrow Y, r) = G_2(X \rightarrow Y, r)/|r|.$$

- G_3: considers the minimum number of tuples in r to be deleted for the FD to hold. Formally: $G_3(X \rightarrow Y, r) = |r| - \max\{|s||s \subseteq r \wedge s \models X \rightarrow Y\}$. The related scaled measurement is defined as g_3 as $g_3(X \rightarrow Y, r)p = G_3(X \rightarrow Y, r)/|r|$.

We can introduce here the definition of approximate functional dependency AFD as:

Definition 4 (Approximate Functional Dependency) Let r be a relationship over the relational schema R: let $X, Y \subseteq R$ be sets of attributes of R. Relation r fulfills the functional dependency $X \overset{\varepsilon}{\rightarrow} Y$ (written as $r \models X \overset{\varepsilon}{\rightarrow} Y$) if $G_3(X \rightarrow Y, r) \leqslant \varepsilon$, where ε is the maximum acceptable error defined by the user.

With AFDs, we can express concepts such as *"for each patient with a given symptom the received treatment does not usually change"*:

$$Patient, \ Symptom \overset{\varepsilon}{\rightarrow} Treatment.$$

Among the several AFDs that can be identified over a relation r, the minimal AFD is of particular interest, as many other AFDs can then be derived from the minimal one. We thus define the minimal AFD as follows:

Definition 5 (Minimal AFD) Given an AFD over r, we define $X \overset{\varepsilon}{\rightarrow} Y$ to be minimal for r if $r \models X \overset{\varepsilon}{\rightarrow} Y$ and $\forall X' \subset X$ we have that $r \not\models X' \overset{\varepsilon}{:\rightarrow} Y$.

3.4.3 Approximate Temporal Functional Dependencies (ATFD)

We now introduce the concept of ATFD, relying on two different kind of temporal grouping, both belonging to the class *Pure Temporally Grouping*.

According to the taxonomy previously defined, we consider here pure temporally grouping TFDs of the form $[r, t - Group]X \rightarrow Y$, where $t - Group$ consists of *granularity (Gran) or sliding window (SW) grouping*.

We recall that a temporal *granularity* is a partition of a temporal domain in indivisible non-overlapping groups, i.e., granules, of time points: minutes, hours, days, months, years as well as working days are *granularities* [49].

Definition 6 (Grouping by Gran(i)) Two tuples $t_1, t_2 \in r$ belong to the same temporal group Gran(i) iff $t_1[VT], t_2[VT] \in$ Gran(i) where Gran(i) is the i^{th} granule of granularity Gran.

A sliding window $SW(i, k)$ includes all the time points in interval $[i \ldots .i + k$ -1]. Thus, once we fix the length of the SW over relation r (i.e., k in the example), every SW over r will feature that length, and will—at most—include k elements (if relation r has tuples for all the time points of interval $[i \ldots i + k$ -1]).

Definition 7 (Grouping by $SW(i, k)$) Two tuples $t_1, t_2 \in r$ belong to the same sliding window $SW(i, k)$ iff $t_1[VT], t_2[VT] \in [i \ldots i + k$ -1].

Before introducing the concept of ATFD, let us consider a new error measure, namely G_4, we shall use for approximate temporal functional dependencies. G_4 considers the minimum number of tuples in r which must be modified for the plain TFD to hold on all the tuples of r. In the following, in the presence of a FD such as $X \rightarrow Y$, we assume to modify values only for the Y attributes.

$G_4([r, t - Group]X \rightarrow Y, r) = \min\{|s| \ |s \subseteq r, ((r - s) \cup w) \models [r, t - Group]X \rightarrow Y|\}$ where the set w is the minimal one for which the following formula holds:

$\forall t \in s(\exists t' \in w(t[U - Y] = t'[U - Y] \wedge t[VT] = t'[VT]))$.

The related scaled measurement g_4 is defined as $g_4(X \rightarrow Y, r) = g_4(X \rightarrow Y, r)/|r|$

We define the ATFD with granularity grouping as:

Definition 8 (ATFD with Gran Grouping) Let r be a relationship over the relational schema R with attributes $U \cup \{VT\}$: let $X, Y \subseteq R$ be attributes of R. Let Gran be the reference granularity $[r, Gran]X \overset{\varepsilon}{\rightarrow} Y$ holds on relation r iff the introduced error $g_3([r, Gran]X \rightarrow Y, r) \leq \varepsilon$ is less than the given threshold ε.

Definition 9 (Minimal ATFD with Gran Grouping) An ATFD in the form of $[r, Gran]X \overset{\varepsilon}{\rightarrow} Y$ is said to be minimal for r iff $r \models [r, Gran]X \overset{\varepsilon}{\rightarrow} Y$ and $\forall X' \subset X$ we have that $r \not\models [r, Gran]X' :\overset{\varepsilon}{\rightarrow} Y$.

We define the ATFD with sliding window grouping as:

Definition 10 (ATFD with SW Grouping) Let r be a relationship over the relational schema R with attributes $U \cup \{VT\}$: let $X, Y \subseteq U$ be attributes of R. Let $\{i \ldots i + k - 1\}$ be a sliding window (SW) of length k. The approximate temporal functional dependency $[r, \{i \ldots i + k - 1\}]X \overset{\varepsilon}{\rightarrow} Y$ holds on relation r iff the introduced error $g_4([r, \{i \ldots i + k - 1\}]X \rightarrow Y, r) \leq \varepsilon$ is lower than the given threshold ε.

Definition 11 (Minimal ATFD with SW Grouping) Given an ATFD over $[r, \{i \ldots i + k - 1\}]$, we define $X \overset{\varepsilon}{\rightarrow} Y$ to be minimal for r iff $r \models [r, \{i \ldots i + k - 1\}]X \overset{\varepsilon}{\rightarrow} Y$ and $\forall X' \subset X$ we have that $r \not\models [r, \{i \ldots i + k - 1\}]X' \overset{\varepsilon}{\rightarrow} Y$.

The study reported in [50] represents the use of previous definitions. The authors applied these in two different clinical domains: the first one referred to psychiatry, collecting data about contacts between patients and psychiatrists, psychologists, and social workers; the second was pharmacovigilance, collecting data about drug administrations and adverse reactions.

The first domain concerned about the Verona Psychiatric Case Register (PCR). The National Health Service in trust with the University of Verona offers a public Community-based Psychiatric Service (CPS), providing psychiatric care to mentally ill as well as psychological care and responses to social needs. Data about patients are collected in the information system PCR, which has recorded information about patients' accesses to this service since 1979. PCR contained patients' personal data, patients' medical record, contact information, records education, employment, professional status, type of accommodation, and marital status. PCR is used as a basis to evaluate the direct management costs for groups of patients, and to monitor the effects coming from changes in resources, organization, and needs. The clinical purposes include monitoring of patients to plan future contacts at regular time intervals, and providing clinicians with reports about admissions and contacts for every patient in a given time period. These temporal data can then be used by psychiatrists, e.g., to identify the number of contacts in different time periods with respect to different factors such as age, diagnosis. A meaningful example discussed in this context was: $[133days]HealthStructure \rightarrow ContactType$. As second medical domain, Pharmacovigilance (PhV) collects, analyzes, and prevents adverse reactions induced by drugs (ADR). The spontaneous reporting of ADRs identifies unexpected reactions and informs the regulating authority about them. This practice is valuable, provides early warnings, and requires limited economic and organizational resources. It also has the advantage of covering every drug on the market and every category of patient. PhV considers possible relationships between one or more adverse reactions and one or more drugs, mainly focusing on unknown or completely undocumented relationships. Reports suggest a cause-effect link among ADRs and drugs. Each report includes patient's information and the description of the occurred adverse reaction. These temporal data are used to investigate any cause-effect relationship among drugs and reaction(s) in different time periods, or according to the time frame of the exposure. A meaningful example discussed in this context was: $[30days]Drug, AdverseReaction \rightarrow Outcome$.

4 Conclusions

In this chapter we analyzed some of the aspects of Data Mining in medicine. In particular, at the beginning we focused on Deep learning and its applications in different clinical fields. Assuming that there are many Data Mining methods from which one can chose to mine the emerging medical databases and there is no an absolute data mining method to solve the issues in the healthcare data sets, in this chapter we have reviewed most popular ones, and gave some pointers where they have been applied. Then we analyzed in a more technical way the Temporal Data Mining aspects, introducing the main concepts of temporal abstractions, interval-based temporal patterns, association rules, and functional dependencies.

References

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, pp. 37–37, 1996.
[2] N. Jothi, W. Husain, *et al.*, "Data mining in healthcare–a review," *Procedia Computer Science*, vol. 72, pp. 306–313, 2015.
[3] O. Maimon and L. Rokach, "Introduction to knowledge discovery and data mining," in *Data mining and knowledge discovery handbook*, pp. 1–15, Springer, 2009.
[4] R. Bellazzi and B. Zupan, "Predictive data mining in clinical medicine: current issues and guidelines," *International journal of medical informatics*, vol. 77, no. 2, pp. 81–97, 2008.
[5] C. Robert, "Machine learning, a probabilistic perspective," 2014.
[6] J.-G. Lee, S. Jun, Y.-W. Cho, H. Lee, G. B. Kim, J. B. Seo, and N. Kim, "Deep learning in medical imaging: general overview," *Korean journal of radiology*, vol. 18, no. 4, pp. 570–584, 2017.
[7] S. K. Pandey and R. R. Janghel, "Recent deep learning techniques, challenges and its applications for medical healthcare system: A review," *Neural Processing Letters*, pp. 1–29, 2019.
[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.
[9] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014.
[10] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2017.
[11] A. A. A. Setio, F. Ciompi, G. Litjens, P. Gerke, C. Jacobs, S. J. Van Riel, M. M. W. Wille, M. Naqibullah, C. I. Sánchez, and B. van Ginneken, "Pulmonary nodule detection in CT images: false positive reduction using multi-view convolutional networks," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1160–1169, 2016.
[12] H. R. Roth, L. Lu, J. Liu, J. Yao, A. Seff, K. Cherry, L. Kim, and R. M. Summers, "Improving computer-aided detection using convolutional neural networks and random view aggregation," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1170–1181, 2015.
[13] Q. Dou, H. Chen, L. Yu, L. Zhao, J. Qin, D. Wang, V. C. Mok, L. Shi, and P.-A. Heng, "Automatic detection of cerebral microbleeds from MR images via 3D convolutional neural networks," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1182–1195, 2016.
[14] K. Sirinukunwattana, S. E. A. Raza, Y.-W. Tsang, D. R. Snead, I. A. Cree, and N. M. Rajpoot, "Locality sensitive deep learning for detection and classification of nuclei in routine colon

cancer histology images," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1196–1206, 2016.

[15] G. Currie, K. E. Hawk, E. Rohren, A. Vial, and R. Klein, "Machine learning and deep learning in medical imaging: Intelligent imaging," *Journal of Medical Imaging and Radiation Sciences*, vol. 50, p. 477–487, Dec 2019.

[16] A. T. Kharroubi and H. M. Darwish, "Diabetes mellitus: The epidemic of the century," *World journal of diabetes*, vol. 6, no. 6, p. 850, 2015.

[17] T. Y. Wong, C. M. G. Cheung, M. Larsen, S. Sharma, and R. Simó, "Diabetic retinopathy," *Nature Reviews Disease Primers*, vol. 2, 2016.

[18] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *Jama*, vol. 316, no. 22, pp. 2402–2410, 2016.

[19] M. D. Abràmoff, P. T. Lavin, M. Birch, N. Shah, and J. C. Folk, "Pivotal trial of an autonomous ai-based diagnostic system for detection of diabetic retinopathy in primary care offices," *NPJ Digital Medicine*, vol. 1, no. 1, p. 39, 2018.

[20] N. Harbeck, F. Penault-Llorca, J. Cortes, M. Gnant, N. Houssami, P. Poortmans, K. Ruddy, J. Tsang, and F. Cardoso, "Breast cancer," *Nature Reviews Disease Primers*, vol. 5, Sep 2019.

[21] B. E. Bejnordi, M. Veta, P. J. Van Diest, B. Van Ginneken, N. Karssemeijer, G. Litjens, J. A. Van Der Laak, M. Hermsen, Q. F. Manson, M. Balkenhol, *et al.*, "Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer," *Jama*, vol. 318, no. 22, pp. 2199–2210, 2017.

[22] J. Wang, X. Yang, H. Cai, W. Tan, C. Jin, and L. Li, "Discrimination of breast cancer with microcalcifications on mammography by deep learning," *Scientific reports*, vol. 6, no. 1, pp. 1–9, 2016.

[23] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, p. 115, 2017.

[24] M. H. Jafari, N. Karimi, E. Nasr-Esfahani, S. Samavi, S. M. R. Soroushmehr, K. Ward, and K. Najarian, "Skin lesion segmentation in clinical images using deep learning," in *2016 23rd International conference on pattern recognition (ICPR)*, pp. 337–342, IEEE, 2016.

[25] Y. Cheng, F. Wang, P. Zhang, and J. Hu, "Risk prediction with electronic health records: A deep learning approach," in *Proceedings of the 2016 SIAM International Conference on Data Mining*, pp. 432–440, SIAM, 2016.

[26] A. Avati, K. Jung, S. Harman, L. Downing, A. Ng, and N. H. Shah, "Improving palliative care with deep learning," *BMC medical informatics and decision making*, vol. 18, no. 4, p. 122, 2018.

[27] A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, *et al.*, "Scalable and accurate deep learning with electronic health records," *NPJ Digital Medicine*, vol. 1, no. 1, p. 18, 2018.

[28] J. A. Golden, "Deep learning algorithms for detection of lymph node metastases from breast cancer: helping artificial intelligence be seen," *Jama*, vol. 318, no. 22, pp. 2184–2186, 2017.

[29] A. R. Post, A. N. Sovarel, and J. H. Harrison Jr, "Abstraction-based temporal data retrieval for a clinical data repository," in *AMIA Annual Symposium Proceedings*, vol. 2007, p. 603, American Medical Informatics Association, 2007.

[30] C. Combi, M. Mantovani, and P. Sala, "Discovering quantitative temporal functional dependencies on clinical data," in *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 248–257, IEEE, 2017.

[31] A. Shknevsky, Y. Shahar, and R. Moskovitch, "Consistent discovery of frequent interval-based temporal patterns in chronic patients' data," *Journal of biomedical informatics*, vol. 75, pp. 83–95, 2017.

[32] R. Moskovitch and Y. Shahar, "Fast time intervals mining using the transitivity of temporal relations," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 21–48, 2015.

[33] C. Combi, E. Keravnou-Papailiou, and Y. Shahar, *Temporal information systems in medicine*. Springer Science & Business Media, 2010.
[34] R. Moskovitch and Y. Shahar, "Classification-driven temporal discretization of multivariate time series," *Data Mining and Knowledge Discovery*, vol. 29, no. 4, pp. 871–913, 2015.
[35] R. Moskovitch and Y. Shahar, "Classification of multivariate time series via temporal abstraction and time intervals mining," *Knowledge and Information Systems*, vol. 45, no. 1, pp. 35–74, 2015.
[36] Y. Shahar, "A framework for knowledge-based temporal abstraction," *Artificial intelligence*, vol. 90, no. 1-2, pp. 79–133, 1997.
[37] Y. Shahar and M. A. Musen, "Knowledge-based temporal abstraction in clinical domains," *Artificial intelligence in medicine*, vol. 8, no. 3, pp. 267–298, 1996.
[38] E. Sheetrit, N. Nissim, D. Klimov, and Y. Shahar, "Temporal probabilistic profiles for sepsis prediction in the ICU," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2961–2969, 2019.
[39] S. Concaro, L. Sacchi, C. Cerra, P. Fratino, and R. Bellazzi, "Mining healthcare data with temporal association rules: Improvements and assessment for a practical use," in *Conference on Artificial Intelligence in Medicine in Europe*, pp. 16–25, Springer, 2009.
[40] L. Sacchi, C. Larizza, C. Combi, and R. Bellazzi, "Data mining with temporal abstractions: learning rules from time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 217–247, 2007.
[41] R. Bellazzi, C. Larizza, and A. Riva, "Temporal abstractions for interpreting diabetic patients monitoring data," *Intelligent Data Analysis*, vol. 2, no. 1–4, pp. 97–122, 1998.
[42] J. F. Allen, "Towards a general theory of action and time," *Artificial intelligence*, vol. 23, no. 2, pp. 123–154, 1984.
[43] C. Combi and A. Sabaini, "Extraction, analysis, and visualization of temporal association rules from interval-based clinical data," in *Conference on Artificial Intelligence in Medicine in Europe*, pp. 238–247, Springer, 2013.
[44] M. Mantovani, C. Combi, and M. Zeggiotti, "Discovering and analyzing trend-event patterns on clinical data," *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 1–10, 2019.
[45] R. Bellazzi, C. Larizza, P. Magni, and R. Bellazzi, "Temporal data mining for the quality assessment of hemodialysis services," *Artificial intelligence in medicine*, vol. 34, no. 1, pp. 25–39, 2005.
[46] C. Combi, A. Montanari, and P. Sala, "A uniform framework for temporal functional dependencies with multiple granularities," in *International Symposium on Spatial and Temporal Databases*, pp. 404–421, Springer, 2011.
[47] C. Combi, A. Montanari, and G. Pozzi, "The T4SQL temporal query language," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 193–202, ACM, 2007.
[48] J. Kivinen and H. Mannila, "Approximate inference of functional dependencies from relations," *Theoretical Computer Science*, vol. 149, no. 1, pp. 129–149, 1995.
[49] C. Combi, M. Franceschet, and A. Peron, "Representing and reasoning about temporal granularities," *Journal of Logic and Computation*, vol. 14, no. 1, pp. 51–77, 2004.
[50] C. Combi, M. Mantovani, A. Sabaini, P. Sala, F. Amaddeo, U. Moretti, and G. Pozzi, "Mining approximate temporal functional dependencies with pure temporal grouping in clinical databases," *Computers in biology and medicine*, vol. 62, pp. 306–324, 2015.

Recommender Systems

Shuai Zhang, Aston Zhang, and Lina Yao

1 Introduction to Recommender Systems

Recommender system (RS) seeks to estimate and predict users' preferences for products/services by filtering from a huge pool of information base with patterns/rules discovered from data usage history [23]. It is at the core of many online services such as social networking sites, video streaming services, and online shopping sites we interact with. For example, Amazon uses recommendation engines to suggest products or goods that users might buy [30]. YouTube tailors the recommendation video list based on users' tastes [22]. Facebook recommends friends and posts that might interest end users. Recommendation is also at the core of the Netflix products. To improve the recommendation quality, it hosted the famous Netflix Prize competition [31], which popularized the research on recommender systems. Microsoft also uses recommender systems to enhance the user experience while using XBox [32]. It is seen that recommender systems are now pervasive in our daily life, and its importance cannot be overemphasized.

The prevalence of recommender systems is mainly owning to the enormous collection of options provided by online platforms. However, with the abundant resources available comes the information overload problem. This is when recommender systems come into play. By drawing and learning from huge datasets, the system can capture users' interests so as to pinpoint their preferred items accurately. The benefits of employing recommender systems are manifold: on the one hand, it can enhance customer satisfaction/delight and improve engagement by providing

S. Zhang (✉) · L. Yao
University of New South Wales, Sydney, Australia
e-mail: shuai.zhang@inf.ethz.ch; lina.yao@unsw.edu.au

A. Zhang
Amazon, Bellevue, WA, USA
e-mail: az@astonzhang.com

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_28

personalized recommendations based on their preferences and interests. On the other hand, it has been proven to be an effective tool to drive high conversion rate and customer retention rate, as a result, leading to revenue increase.

1.1 Concepts and Notations

Some important terms that are widely used throughout this chapter are listed and explained below.

Items/Users Items refer to the objects that are recommended. It can be movies, songs, games, news, books, blogs, etc. Users are the people to whom the items are recommended.

Explicit/Implicit Feedback Users give feedback on items to express their preferences and interests. Explicit feedback directly show how a user rates an item, such as rating scores from 1 to 5. Implicit feedback such as clicks, watches, and purchase only serve as a proxy that provides us heuristics about how a user likes an item.

Collaborative Filtering The system Tapestry first coined the term collaborative filtering [39]. Since then, it has been widely used to represent recommender systems. The basic idea of CF is that users will act on items similarly if they have similar behaviors in the past. Most recommendation methods such as neighborhood methods and factorization-based methods that utilize the collaborative filtering idea can be called as collaborative filtering techniques.

The goal of recommender systems is to recommend items to users, from which many tasks are derived. We can formulate the recommendation problem as a regression, classification, ranking, or even sequence modeling problem. For example, estimating the exact rating a user might give to an item is a regression problem; predicting whether an item will be clicked or not belongs to the classification category; generating a ranked list of items for a user can be solved as a learning to rank problem; sequence modeling models come into play if we need to take the sequential patterns of user behaviors into account.

In formal, suppose we have a corpus of M users and N items, which forms an interaction matrix or utility matrix $X \in \mathbb{R}^{M \times N}$. Let $\mathcal{U}$ denote the user set and $\mathcal{I}$ denote the item set. In this matrix, rows correspond to users and columns to items. Generally, this matrix is very sparse, and each entry of this matrix displays users' feedback such as ratings or like/dislikes to the items. Let X_{u*} denote the preferences of user u toward all items. Let X_{*i} denote the feedback (ratings) from all users for item i. These notations will be used throughout the chapter.

2 Recommendation Techniques

In this section, we will introduce some widely used recommendation techniques, including classic solutions and recent advances, and discuss their advantages and weakness.

2.1 Non-personalized and Lightly Personalized Recommendations

A non-personalized recommendation approach that makes the same recommendations for all users is the most basic form of recommender system. Despite its non-personalization, it can be remarkably effective for cases such as common displays in online communities (e.g., reddit) or recommendations for new users for whom we know nothing about. Lightly personalized recommender systems refer to methods that utilize limited information such as user profiles to infer their interests roughly so as to make weakly personalized recommendations.

Recommending based on item popularity is one of the most widely used non-personalized recommendation approaches. We can identify the most popular items by counting the number of likes/views/purchases, etc. If explicit ratings are available, we can also rank the items based on the mean of the ratings (e.g., top rated movies in IMDB). This method is simple yet computationally efficient. It is worth noting that several settings can influence the recommendation performance largely, such as the period for which the popularity is calculated and the interaction types taken into consideration.

Lightly personalized recommendation is a small step toward personalization that could loosely personalize the recommendation list based on certain types of side information such as demographics. The motivation behind it is that users' preferences can be vaguely identifiable with their profiles such as age, gender, race/ethnicity, financial status, location, etc. It is straightforward to break down summary statistics by demographic. For example, tastes on movies can be quite different for people in different ages and stages. Obviously, getting the data about users is critical in this method. As such, it is common to see that some online platforms require new users to take a survey before accessing the services.

Notwithstanding the usefulness of non-personalized or lightly personalized recommendations, their demerits are conspicuous, that is, the recommended items may not satisfy user's interest. That is also why personalized recommender systems start to arise. The following text will be centered on personalized recommendation.

2.2 Neighborhood Methods

There are two standard nearest neighborhood recommendation algorithms: user-based collaborative filtering and item-based collaborative filtering [1].

2.2.1 User-Based Collaborative Filtering

User-based CF aims to find the users who have similar taste (neighbors) as the target user and then recommends items based on the neighbor's interaction behavior. The similarity calculation is based on interaction behaviors. Various similarity measures such as cosine similarity, Pearson's correlation, and Jaccard similarity are viable. Formally, let $sim(X_{u*}, X_{v*})$ denote the similarity between user u and user v. The cosine similarity is defined as below:

$$sim(X_{u*}, X_{v*}) = \frac{X_{u*} \cdot X_{v*}}{\parallel X_{u*} \parallel \parallel X_{v*} \parallel}. \tag{1}$$

Pearson correlation is used to find the linear correlation between two vectors, ranging from -1 to $+1$, with -1 indicating negative relation, 0 representing no relation, and +1 representing high positive correlation. It is defined as

$$sim(X_{u*}, X_{v*}) = \frac{\sum_{i=1}^{N}(X_{ui} - \bar{X_{u*}})(X_{vi} - \bar{X_{v*}})}{\sqrt{\sum_{i=1}^{N}(X_{ui} - \bar{X_{u*}})^2 \sum_{i=1}^{N}(X_{vi} - \bar{X_{v*}})^2}}, \tag{2}$$

where $\bar{X_{u*}}$ represents the average rating of user u.

Afterward, it selects the top K similar users and takes the weighted average of recommendation scores from these K users. To avoid user bias that some users tend to give high scores and some tend to give low scores, users' average ratings are considered. As such, the predicted score is calculated as follows:

$$X_{uj} = \bar{X_{u*}} + \frac{\sum_{k=1}^{K} sim(X_{u*}, X_{k*})(X_{kj} - \bar{X_{k*}})}{\sum_{k=1}^{K} sim(X_{u*}, X_{k*})}. \tag{3}$$

2.2.2 Item-Based Collaborative Filtering

Item-based collaborative filtering applies the same idea. Instead of computing the users similarity, it considers items similarity. Let $sim(X_{*i}, X_{*j})$ denote the similarity between item i and item j. Intuitively, the similarity is measured by observing all the users who have rated both the items. The prediction function of item j for target user u is as follows:

$$X_{uj} = \frac{\sum_{k=1}^{K} sim(X_{*j}, X_{*k}) X_{uk}}{\sum_{k=1}^{K} sim(X_{*j}, X_{*k})}. \tag{4}$$

Item-based CF is more stable and faster in system where there are more users than items. Item similarity matrix can usually be calculated offline as the ratings received by an item do not change quickly (e.g., a recognized good movie usually gets higher rating scores). So it does not need to be recomputed frequently.

2.3 Factorization-Based Methods

Factorization-based approaches (or latent factor models) aim to factorize the interaction matrix with either explicit ratings or implicit feedback into low-dimensional rectangular matrices. These methods enjoy higher flexibility and efficiency than neighborhood-based algorithms.

2.3.1 Matrix Factorization

Matrix factorization [2] method decomposes the user–item interaction matrix into two lower-dimensional matrices for users and items, respectively.

Let $P \in \mathbb{R}^{M \times k}$ represent the user matrix and $Q \in \mathbb{R}^{N \times k}$ represent the item matrix. Each row of P represents the latent factors for describing user's interests and preferences. Each row of the item matrix Q describes items' characteristics. The core idea of matrix factorization is to approximate the interaction matrix with the inner product of P and Q:

$$X \approx P Q^T. \tag{5}$$

To learn the user and item matrices, we can minimize the following mean squared error (MSE) if the goal is to recover the explicit ratings:

$$\min \sum_{(u,i) \in \mathcal{K}} (X_{ui} - P_u Q_i^T)^2 + \|P\|_F^2 + \|Q\|_F^2, \tag{6}$$

where $\mathcal{K}$ is the observed ratings. The last two terms are used to regularize the model parameters. This optimization problem can be efficiently solved with methods such as stochastic gradient descent.

For implicit feedback, a pairwise loss Bayesian personalized ranking (BPR) loss [6] can be used. The BPR loss is defined as follows:

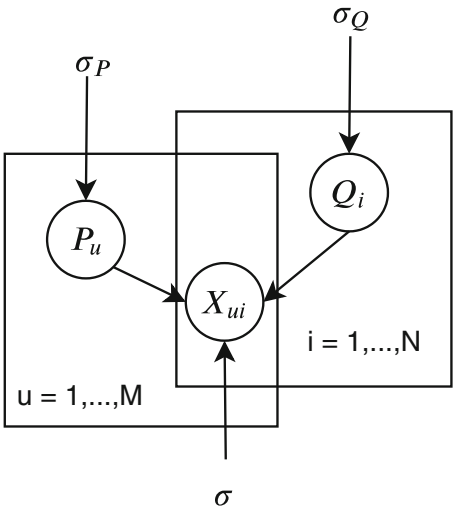


Fig. 1 Graphical model for probabilistic matrix factorization

$$\min \left(- \sum_{(u,i,j) \in \mathcal{K}} \ln \left(\sigma \left(P_u Q_i^T - P_u Q_j^T \right) \right) + \| P \|_F^2 + \| Q \|_F^2 \right). \tag{7}$$

In this loss function, $\mathcal{K}$ is composed of both observed and unobserved feedback. Here, i denotes the item that u likes and j is the item that u has never interacted with.

We have discussed the biases of user preferences in neighborhood methods. These biases should also be captured in matrix factorization. To this end, we can rewrite the scoring function as

$$X_{ui} \approx P_u Q_i^T + b_i + b_u + \mu, \tag{8}$$

where μ is the overall average rating; b_u and b_i indicate the observed deviations of user u and item i. The objective function should be reformulated accordingly.

Owing to the flexibility of matrix factorization method, additional input sources such as implicit feedback, temporal dynamics (SVD++ [3]), and social networks (SoRec [4]) can also be integrated.

The matrix factorization techniques can also be interpreted probabilistically [5]. For example, we can model the rating as a distribution parametrized by item and user latent features (Fig. 1). Assuming ratings are normally distributed:

$$p(X|P, Q, \sigma^2) = \prod_{i=1}^{M} \prod_{j=1}^{N} \left[\mathcal{N} \left(X_{ij} | P_i Q_j^T, \sigma^2 \right) \right]^{I_{ij}}, \tag{9}$$

where the mean is determined by user and item latent factors, and the variance σ^2 is used to model the noise of ratings. We define the indicator I_{ij} to be 1 if X_{ij} is known (i.e., user i has rated movie j) and 0 otherwise. We assume that users and items follow the zero-mean normal distribution with spherical Gaussian priors.

$$p(P|\sigma_P^2) = \prod_{i=1}^{M} \mathcal{N}(P_i|0, \sigma_P^2\mathbf{I}),\ p(Q|\sigma_Q^2) = \prod_{i=1}^{N} \mathcal{N}(Q_i|0, \sigma_Q^2\mathbf{I}). \tag{10}$$

This probabilistic model can be solved with expectation maximization (EM) or gradient descent algorithms. It is worth noting that eventually the optimization process of EM is identical to MSE minimization.

2.3.2 Factorization Machines

Factorization machine, as a generic method, can be used for regression, classification, and ranking tasks. It is essentially an extension of the matrix factorization algorithm and is powerful in dealing with large-scale sparse datasets and automatically modeling the feature interactions. As such, it has been widely used in fields such as products/advertisements recommendations and click-through predictions.

Let $x^{(i)} \in \mathbb{R}^D$ represent the feature vector and $y^{(i)}$ indicate the corresponding target. $x^{(i)}$ can be comprised of the one-hot representations of user/item identities and many other features such as user profiles, latest rated movies by the user, and so on. Generally, the input feature size can be very large and sparse. Label $y^{(i)}$ can represent the exact rating that the user gave to the item or a binary label indicating whether the item is clicked/liked/bought by the user or not.

Theoretically, an FM can model high degree of feature interactions, but 2-way FM is usually employed for efficiency and stability concerns. The scoring function of a 2-way factorization machines is as follows:

$$\hat{y} = w_0 + \sum_{j=1}^{D} \mathbf{w}_i x_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} < V_i, V_j > x_i x_j, \tag{11}$$

where $w_0 \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^D$, and $V \in \mathbb{R}^{D\times k}$ are the model parameters to be learned. Same as matrix factorization, k is the dimension of latent factors. $< \cdot, \cdot >$ denotes the dot product of vectors. This model will degrade to matrix factorization when x_i only contains the user and item one-hot identifiers.

The computation complexity of last term of Eq. 11 in a straightforward way is $O(kD^2)$, which is very expensive. Fortunately, the computation time can be reduced to linear time $O(kD)$ by expanding and reorganizing as follows:

$$\begin{aligned}
&\sum_{i=1}^{D}\sum_{j=i+1}^{D} < V_i, V_j > x_i x_j \\
&= \frac{1}{2}\sum_{i=1}^{D}\sum_{j=1}^{D} < V_i, V_j > x_i x_j - \frac{1}{2}\sum_{i=1}^{D} < V_i, V_i > x_i x_i \\
&= \frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{D} V_{j,f} x_i\right)\left(\sum_{j=1}^{D} V_{j,f} x_j\right) - \sum_{i=1}^{D} V_{i,f}^2 x_i^2\right) \\
&= \frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{D} V_{i,f} x_i\right)^2 - \sum_{i=1}^{D} V_{i,f}^2 x_i^2\right).
\end{aligned} \tag{12}$$

By doing so, the complexity is linear to the number of nonzero elements for sparse inputs. For model training, a variety of loss functions such as MSE, cross-entropy loss, and BPR loss [6] are viable.

2.3.3 Collaborative Metric Learning

Both MF and FM model the interactions between users and items with inner product. However, inner product does not satisfy the triangle inequality, which might limit the expressiveness of the recommendation models. To alleviate the issue, researchers explore using distance functions (e.g., Euclidean distance [7], hyperbolic distance [40]) to replace the inner product. In the inference stage, items that are close to the user are recommended.

Collaborative metric learning [7] is such a representative model. It assumes that the positions of users and items are represented by $P \in \mathbb{R}^{M\times k}$ and $Q \in \mathbb{R}^{N\times k}$, and the distance between user u and item i is measured by

$$d(u, i) = \|P_u - Q_i\|_2^2. \tag{13}$$

A max-margin triplet loss is usually used for model optimization. The goal of the loss is to ensure the distance between a user and the item she likes to be smaller than that between the user and the item that she dislikes.

$$L = \sum_{(u,i)\in\mathcal{S}}\sum_{(u,j)\in\mathcal{S}'} \max(0, d(u, i) + \lambda - d(u, j)), \tag{14}$$

where set $\mathcal{S}$ is made of users and their liked items and set $\mathcal{S}'$ contains users and their disliked items. Regularization (e.g., norm clipping) is usually used on P and Q to prevent the data points spread too widely.

2.4 Modeling Sequences in Recommendation

Intuitively, there usually exist sequential patterns in user behaviors and interaction trajectories. Users usually have long-term and short-term interests. So far, we only consider users' long-term taste and all short-term preferences are ignored. However, users' short-term intents play a critical role in users' decisions [8]. For example, if a user bought a digital single lens reflex (DSLR), she will probably buy a camera Lens shortly. Knowing this pattern is important for making satisfying recommendations. The capability to simultaneously model both long-term and short preferences is the key to sequence-aware recommendation models.

Suppose each user u is associated with a sequence of items $S^u = (S_1^u, \dots S_{|S^u|}^u)$, where S_t^u represents the item user u interacted with at time step t that does not need to be absolute time but just an indicator of sequence order. The goal of sequence-aware recommendation is to predict the next item that a user will interact with.

The model we will introduce is a variant of collaborative metric learning, called personalized ranking metric embedding (PRME) [9], which considers both user general and transient intents. Let $Q \in \mathbb{R}^{N \times k}$ denote item embeddings. To model the transient interest, the model aims to make adjacent items in the sequence close to one another. Therefore, the following distance shall be minimized:

$$d(S_{t-1}^u, S_t^u) = \| Q_{S_{t-1}^u} - Q_{S_t^u} \|_2^2, \tag{15}$$

where S_{t-1}^u and S_t^u are adjacent items and S_t^u is the target item. The motivation behind this is that if two items are interacted subsequent, they are more likely to be similar.

The general taste module has the same form as collaborative metric learning. The goal is to minimize the distance between user u and the target item.

$$d(u, S_t^u) = \| P_u - V_{S_t^u} \|_2^2, \tag{16}$$

where $P \in \mathbb{R}^{M \times k}$ is the user embeddings and $V \in \mathbb{R}^{N \times k}$ is the item embeddings.

The final recommendation score is determined by the weighted summation of these two distances:

$$\omega \cdot d(S_{t-1}^u, S_t^u) + (1 - \omega) \cdot d(u, S_t^u), \tag{17}$$

where ω determines the proportion of contributions of short-term and long-term interests.

2.5 Neural Architectures for Recommender Systems

In recent years, deep neural networks have achieved tremendous success in a number of fields such as computer vision, natural language processing, speech recognition, and so on [10]. A number of deep learning techniques such as convolutional neural networks, recurrent neural networks, generative adversary networks, graph neural networks, attention networks, and deep reinforcement learning are gaining popularity in both industry and academia.

In the meantime, deep neural networks have been revolutionizing the recommendation structures as well [11]. A large amount of deep-learning-based recommendation architectures are proposed these years. It has also been demonstrated to be especially useful in real-world recommendation scenarios, and a number of companies are building their recommender systems with deep neural networks. The major advantages are: First, deep learning is capable of modeling complex data and learning expressive and high-level representations. Second, deep neural networks are advantageous in modeling sequence data and capturing the hidden sequential patterns. Third, deep neural networks can be trained end-to-end, and they have good composability and flexibility, making the design of more powerful joint models possible. In this section, we will introduce some recent advancements on deep neural-networks-based recommender systems. It is worth noting that this section is highly relevant to the content-based recommender systems that will be introduced later.

2.5.1 From Linear to Nonlinear Recommendation Models

Methods such as MF and FM use linear transformation to model the feature interactions (e.g., interaction between user latent vector and item latent vector). However, the patterns hidden in the interaction data might be extreme complex and intricate. Using nonlinear neural networks can capture the interaction patterns more easily. Here we introduce several popular models that implement this idea.

Neural Collaborative Filtering [12]. To enrich the model expressiveness, this model consists of a multilayered perceptron (MLP) and a generalized matrix factorization. Like matrix factorization, it uses latent vectors to represent each user/item. Formally, let $P \in \mathbb{R}^{M \times k}$ and $U \in \mathbb{R}^{M \times k}$ denote user latent embeddings, and use $Q \in \mathbb{R}^{N \times k}$ and $V \in \mathbb{R}^{N \times k}$ to represent each item.

The input of MLP is the concatenation of P_u and Q_i:

$$\begin{aligned} h_1 &= \alpha_1(W_1 \cdot [P_u, Q_i] + b_1) \\ &\cdots \\ h_{\ell-1} &= \alpha_{\ell-1}(W_{\ell-1} \cdot h_{\ell-2} + b_{\ell-1}) \\ h_\ell(u, i) &= \alpha_\ell(W_\ell \cdot h_{\ell-1} + b_\ell), \end{aligned} \tag{18}$$

where $[\cdot, \cdot]$ denotes the concatenation operation. ℓ is the depth of the MLP. W_*, b_*, and α_* are weight, bias, and activation function. $h_\ell(u, i)$ is the output of the MLP. This component is mainly used to model the complex and nonlinear interactions between users and items.

The input for the generalized MF component is the entry-wise (Hadamard) product of user and item latent factors, and it is defined as

$$o(u, i) = \alpha(W \cdot (U_u \odot V_i)). \tag{19}$$

Afterward, the outputs of two components are concatenated and transformed with a nonlinear layer to get the final prediction score.

$$\hat{X_{ui}} = \alpha(W[h_\ell(u, i), o(u, i)]). \tag{20}$$

The model can be trained with commonly used MSE, BPR loss, or the cross-entropy loss.

Autoencoder for recommendation. An autoencoder is a feed-forward neural network that codes its input to output while learning a hidden representation in the bottleneck layer. It is a useful dimensionality reduction model. It can also be used to reconstruct the interaction matrix. Here, we introduce two models (AutoRec [13] for rating prediction and CDAE [14] for ranking with implicit feedback).

Similar to neighborhood methods, AutoRec can be either user-based or item-based. The input of the item-based AutoRec is the column of the rating matrix. The model consists of the following encoder and decoder:

$$\begin{aligned} Encoder &: h = \alpha_1(W_1 X_{*i} + b_1) \\ Decoder &: o = \alpha_2(W_2 h + b_2), \end{aligned} \tag{21}$$

where W_*, b_*, and α_* are weight, bias, and activation function. o has the same dimensionality as X_{*i}.

The loss function of AutoRec aims to minimize the following reconstruction error:

$$\underset{W_*, b_*}{\operatorname{argmin}} \sum_{i=1}^{M} \|X_{*i} - o\|_O^2 + \lambda(\|\mathbf{W}_*\|_F^2), \tag{22}$$

where $\|\cdot\|_O$ means that only observed ratings are contributed to the gradient backpropagation. With partial observed columns as input, it targets at reconstructing the entire columns. The user-based AutoRec is similar to the item-based AutoRec, but it uses rows instead of columns of the rating matrix as input.

CDAE also employs an autoencoder framework, but it is designed for recommendation with implicit feedback. Simply put, the model architecture is defined as

$$o = \alpha_2(W_2 \cdot \alpha_1(W_1 X_{u*} + U_u + b_1) + b_2), \tag{23}$$

where W_*, b_*, and α_* are weight, bias, and activation function. $U_u \in \mathbb{R}^{M \times k}$ is a user-specific bias. The loss function of CDAE is

$$\min_{W_*, U_*, b_*} \sum_{i=1}^{N} \ell(X_{u*}, o) + \lambda(\|\mathbf{W}*\|_F^2 + \|\mathbf{U}*\|_F^2), \tag{24}$$

where ℓ can be MSE or logistic loss. It is worth noting that instead of masking all unobserved input such as AutoRec, CDAE allows sampled unobserved feedback as input.

2.5.2 Representation Learning with Neural Architectures

Deep neural networks are powerful feature representation tools. They map raw features with a number of neural layers and get an abstraction of the input features in either supervised or unsupervised manner.

Multilayer perceptron is an effective tool for feature representation learning in recommender systems. The model **Wide & Deep** learning [15] proposed by Google is a good example. This model has shown good performance in Google play app store. This model consists of a wide component and a deep component. The wide component is a linear regression model that is helpful for memorization of feature interactions (e.g., co-occurrence of items), while the deep component a multilayer perceptron that could generalize to unseen feature combinations through low-dimensional dense embeddings.

In formal, the input is split into two parts: one for the wide network and the other for the deep network. We denote them with x^{wide} and x^{deep}, respectively. Same as FMs, the features are sparse. For the deep component, we let $V \in \mathbb{R}^{D \times k}$ denote the dense embeddings for the sparse feature inputs x^{deep}. For simplicity, we assume that the input features are made of m fields. After looking up from V with x^{deep} and concatenation, we get

$$h(V, x) = [e_1, e_2, \ldots, e_m]. \tag{25}$$

It is used as the input of the deep component. The final scoring function is defined as

$$\hat{y} = \sigma(W * x^{\text{wide}} + f_{MLP}(h(V, x^{\text{deep}})) + b), \tag{26}$$

where f_{MLP} is the MLP network.

DeepFM [16] replaces the linear part of wide and deep model with factorization machines. Even though linear model is effective for memorization, it is not capable of model direct feature interactions. As introduced in earlier section, FM can model 2-way interactions efficiently. Using FM as a replacement of the wide part enables

it to explicitly model feature interactions but will not incur additional computation cost.

Using DeepFM, the explicit split of features into two parts is no longer necessary, which could extensively reduce the efforts in feature engineering. The scoring function of DeepFM is

$$\hat{y} = \sigma(\hat{y}_{FM}(x) + f_{MLP}(h(V, x))). \tag{27}$$

Item2Vec [33]. Neural networks can also be used for item representations learning. Barkan et al. [33] proposed item2vec to learn item representations in a similar way to the word2vec approach [34]. In item2vec, items can be viewed as words, and the sequences of items a user liked can be seen as sentences. In doing so, the same skip-gram with negative sampling algorithm as that of word2vec can be utilized for item embedding learning.

2.5.3 Sequence-Aware Recommendation with Neural Networks

We introduced the concept of sequence-aware recommendation with a representative model PRME. However, the sequence length in PRME is merely one. With neural networks, we can model longer sequences. Let L denote the length of the sequence. The L embedding vectors form a matrix:

$$E^{(u,t)} = \begin{bmatrix} Q_{S^u_{t-L}} \\ \cdots \\ Q_{S^u_{t-1}} \end{bmatrix}. \tag{28}$$

To learn patterns from this matrix, a number of neural architectures are viable, such as RNN, CNN, and attention networks. In this section, we will introduce a self-attention-based sequential recommendation method [18].

There are three important concepts in an attention network: query, key, and value. For self-attention, all of them are equal to $E^{(u,t)}$. At first, nonlinear transformations are applied on query and key:

$$Q' = \sigma(E^{(u,t)} W_Q) \tag{29}$$

$$K' = \sigma(E^{(u,t)} W_K), \tag{30}$$

where $W_Q \in \mathbb{R}^{k \times k} = W_K \in \mathbb{R}^{k \times k}$ are weight matrices. σ is activation function (usually ReLU). Then, the affinity matrix is calculated as follows:

$$z^{(u,t)} = \text{softmax}\left(\frac{Q' K'^T}{\sqrt{k}}\right), \tag{31}$$

where the output is an $L \times L$ affinity matrix (or attention map). $\sqrt{d}$ is used to scale the dot product attention. Afterward, it weights the value matrix with this attention map. Then it uses mean pooling to aggregate all the L vectors into a single vector.

$$m^{(u,t)} = \frac{1}{L} \sum_{l=1}^{L} z^{(u,t)} E^{(u,t)}. \tag{32}$$

Lastly, we replace the $Q_{S^u_{t-1}}$ in Eq. (15) and train the model in the same way. This attention module could greatly improve the expressiveness of PRME in short-term interest modeling.

2.5.4 Advanced Topics and New Frontiers

In recent years, two topics including graph neural networks [20] and deep reinforcement learning [10] are getting increasing popularity in both academia and industry. Graph neural networks work on graphs and utilize the message passing mechanism for node/graph representation learning. It is natural to apply this technique to recommendation tasks [19] as entities in recommender systems can be organized as graphs. For example, the interaction matrix can be viewed as a bipartite graph between users and items; relations between users can also form a social graph. Deep reinforcement learning is another promising technique for recommender systems. The idea behind reinforcement learning is that an agent will learn from the environment by interacting with it and receiving rewards for performing actions. There are five key concepts, including environment, agent, reward, state, and action. For recommendation task, we can consider the users and items pool as the environment, the recommendation model as agents, clicks/no clicks as rewards, features of users as states, and features of candidature items as actions [21]. However, there are still a number of challenges (e.g., scalability) in these fields that remain to be solved.

In addition, increasing the explainability of recommendations using neural networks is also useful. In many cases, both customers and developers want to know the reason why a specific item is recommended. However, most current models lack this capability. To enhance the explainability, a lot of effort has been made [35]. Readers are referred to [36] for a comprehensive survey.

2.6 Content-Based Recommender Systems

Content-based recommender systems recommend items based on the content (e.g., descriptions, article, videos, etc.) of items and a profile of the user's interests [25]. Content is the essential element in a digital world. Content can be created in many

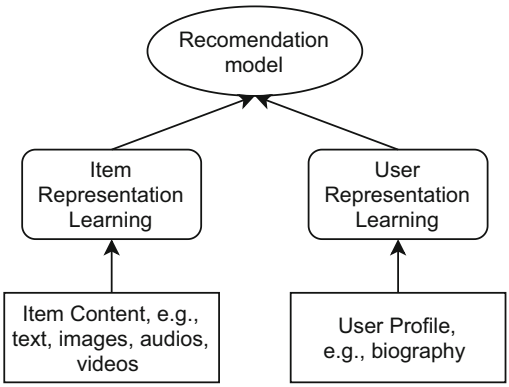


Fig. 2 The framework of content-based recommender systems

different formats. It can be structured tables/graphs or unstructured text, images, audios, videos, etc. To make effective recommendations, it is important for the recommender system to understand the content. Unlike the recommendation methods purely based on the user–item interaction matrix, content-based recommender systems are methods that combine both content and the interactions. In general, content-based recommender systems need a component to learn representations from content and a recommendation module such as MF. Naturally, many collaborative filtering approaches can be integrated as a part of content-based recommender systems. It is also called hybrid recommender systems used frequently in the literature [26, 28]. Figure 2 is a typical framework of content-based recommender systems.

The choice of the methods used for content representation learning is highly dependent on the content format. Early works used tf*idf, decision trees, and linear classifiers to model the content [25]. Nowadays, with the development of neural networks, it is a more common choice to handle the content with deep neural networks. If we have a table of categorical features, using aforementioned methods such as DeepFM and Wide&Deep learning model would be a good fit. Other neural architectures such as convolution neural networks, autoencoder, and transformer [27] can be used for more complex features. Specifically, these methods are especially effective for multimedia data sources such as text [29], image, audio, and even video. For example, convolutional neural etworks with flexible convolution and pooling operations are effective in capturing the spatial and temporal dependencies in images and texts. A number of well-defined CNN architectures such as GoogleNet and ResNet [17, 10] are ready for use. Readers are referred to the survey [11] for more detailed descriptions of deep learning solutions for these tasks.

3 Recommendation Tasks and Applications

Recommender systems are growing more popular mainly due to its usefulness in real-world applications. In this section, we will present some widely studied applications, concerns in industrial-scale recommendation, and a few of open-source toolkits that enable practitioners to get hands-on experience.

3.1 Applications of Recommender Systems

Point-of-Interest Recommendation Point-of-interest (POI) becomes popular with the emergence of location-based social network (LBSN) such as Foursquare,[1] Gowalla, and Facebook Place.[2] On these online platforms, users can check in and share their experiences about the places. The task of POI recommendation is to recommend places for users to visit. It can increase the users' viscosity to the LBSN service provider and help advertising agency to locate potential customers. POI recommendation is a representative application of sequence-aware recommendation systems as users' check-in data usually show strong sequence patterns in terms of time and geography.

Social Recommendation Social media platforms such as Facebook, Twitter, and Instagram connect users with people they are familiar with or business they are interested in. Recommender algorithms in these platforms aim to recommend people to follow, pages to like, and posts to read based on users' previous engagement and usage. A key consideration in social recommendation is the social relations such as friendship, following relationship, and membership. Social relations explicitly show the neighborhood of a user and the trust relations between users that could act as a strong regularization for recommendations. For example, we can force the users' representations to be close if they are friends.

Multimedia Recommendation Multimedia data are ubiquitous in our daily life nowadays. For example, news and blogs usually consist of text, images, and video; YouTube videos have text descriptions and subtitles; music pieces have text lyrics and album cover apart from the music audio resources. Recommending multimedia content requires the recommendation model to properly process these multimedia signals. Extracting content descriptors in these data is a well-established research task. A number of recent advanced techniques in fields such as NLP, CV, and Multimedia might be useful. For example, we can learn text representations with BERT [41], extract visual features with ResNet or Vision Transformers, and so on.

[1] https://foursquare.com/.

[2] https://www.facebook.com/places/.

Other Domains The usage of recommender systems is not limited to the aforementioned domains. Here, we will list a few more cases. For example, recommending games, news, mobile applications, cars, etc., is of great practical use for vendors. Another interesting direction is fashion-aware recommendation that involves recommending fashion-related products (e.g., clothes) by inspecting the fashion elements [37]. In addition, choosing suitable food recipe based on users' health condition [38] and recommending beverage (e.g., wine) based on customers' taste are also possible with some dedicate designs.

3.2 Practice for Industrial-Scale Recommendation

For real-world industrial-level recommender systems, the scale of dataset is usually way larger than the dataset used in academia research and beyond the ability of commonly used model. To address this issue, a two-step process that includes candidate generation and ranking is usually deployed [22]. First, it generates a set of recommendation candidates from the massive corpus with techniques such as matrix factorization. Then, it fine-tunes the candidate set with more detailed inputs with more advanced models. This is a compromise between accuracy and complexity, but it now becomes a common practice in industry. Another important aspect of industrial-level recommendation is feature engineering. Deciding which features are predictive heavily relies on expert's experiences. For some certain recommendation tasks, some specific features or combinations of features are critical for model performances. Additionally, online test is a very important step in evaluating the actual effectiveness of a recommendation model in industry. A/B testing (bucket testing) is one of the most popular online test approaches where two models are deployed to randomly serving visitors for comparison to determine which one performs better.

3.3 Tool Kits for Building Recommender Systems

To become familiar with the concepts and techniques in recommender systems, it is a good idea to get some hands-on experiences. However, implementing a recommender system from scratch is usually troublesome and time-consuming. As such, we collected some open-source recommendation libraries that aim to help us demonstrate or build a simple recommender model easily.

- **MyMediaLite**.[3] It is an open-source recommendation library published in 2011. It supports three programming languages: C#, Clojure, and F#. It provides algorithms on both rating prediction and item ranking tasks.

[3] http://mymedialite.net/index.html.

- **DeepRec**.[4] It is an open-source library for recommendation with deep neural networks. It is a Python library that uses TensorFlow as its backend and addresses tasks such as rating prediction, item ranking, and sequence-aware recommendation.
- **LibRec**.[5] It is a Java library for recommendation. It aims to solve the rating prediction and item ranking tasks. A number of traditional recommendation algorithms are provided.
- **Suprise**.[6] It is a Python toolkit that provides a limited amount of rating prediction models.
- **OpenRec**.[7] OpenRec is also a Python recommendation library. In this library, each recommender is a structured ensemble of reusable modules. However, there are only a few algorithms implemented.

4 Evaluate Recommender Systems

Proper evaluation measures are critical to building a satisfying recommender system. Evaluation is an important step in deciding which recommendation algorithm is the best. For different recommendation tasks, we may need different evaluation measures. Here, we introduce some commonly used ones.

4.1 Evaluation on Recommendation Accuracy

In general, accuracy is the top priority and the major concern of most recommender systems. Here, we summarize several commonly used accuracy measures. We omit the measures used in classification tasks as they are commonplace in other areas.

Root Mean Square Error (RMSE) RMSE is a widely used evaluation measure for measuring the accuracy of ratings prediction. The definition is as follows:

$$\text{RMSE} = \sqrt{\frac{1}{|T|} \sum_{(u,i)\in T} (\hat{X_{ui}} - X_{ui})^2}, \tag{33}$$

where T is the dataset we want to evaluate on, $\hat{X_{ui}}$ denotes the predicted ratings, and X_{ui} is the ground truth. RMSE will give relatively high weight to large errors

[4] https://github.com/cheungdaven/DeepRec.
[5] https://www.librec.net/.
[6] http://surpriselib.com/.
[7] https://openrec.ai/.

as the errors are squared before averaged. It is most useful when large errors are particularly undesirable.

Mean Average Error (MAE) It is also commonplace in measuring the accuracy of rating prediction task. It measures the average magnitude of errors and is defined as follows:

$$\text{MAE} = \frac{1}{|T|} \sum_{(u,i)\in T} |\hat{X_{ui}} - X_{ui}|. \tag{34}$$

Individual differences are weighted equally in MAE.

Recall Recall at n is the proportion of items the user liked found in the top-n recommendation list (recommended items are ranked as a list where higher ranked items are the ones that the user will like the most):

$$\text{Recall@}n = \frac{\text{Number of items that } u \text{ likes among the top-}n \text{ list}}{\text{liked}(u)}. \tag{35}$$

The final result is the average recall over all users. liked(u) is the total number of items that user u liked.

Precision Precision at n is the proportion of recommended items in the top-n list that are liked by the user:

$$\text{Precision@}n = \frac{\text{Number of items that } u \text{ likes among the top-}n \text{ list}}{n}. \tag{36}$$

The overall precision is the average precision over all users.

Mean Average Precision (MAP) It is different from precision in that correctly recommended items in top ranks are prioritized.

$$\text{MAP@}n = \frac{1}{M} \sum_{u=1}^{M} \frac{\sum_{j=1}^{n} \text{Precision@}j \times \mathbf{1}_{\text{rel}}(j)}{\min\{n, \text{liked}(u)\}}, \tag{37}$$

where $\mathbf{1}_{\text{rel}}(j)$ is an indicator function equaling 1 if the item at rank k is liked by the user. Obviously, MAP is a rank-aware evaluation metric as it rewards the system for having the "correct" items higher ranked in the list.

Normalized Discounted Cumulative Gain (NDCG) It is also a rank-aware measure, where positions are discounted logarithmically. The definition of NDCG is as

$$\text{DCG@}n = \sum_{j=1}^{n} \frac{\mathbf{1}_{\text{rel}}(j)}{log_2 j + 1}. \tag{38}$$

And NDCG@n $=$ $\frac{\text{DCG@}n}{\text{IDCG@}n}$ with IDCG@n denoting the DCG for perfect ranked list.

Mean Reciprocal Rank (MRR) It cares about the single highest ranked relevant item, and it calculates the reciprocal of the rank at which the first item is put.

$$\text{MRR} = \frac{1}{M}\sum_{u=1}^{M}\frac{1}{\text{rank}_u}, \tag{39}$$

where rank_u is the rank of the first correctly ranked item for user u.

4.2 Beyond Accuracy

Beyond accuracy, there are many other aspects such as coverage, privacy, diversity, novelty, robustness, scalability, explainability, and freshness that are important for recommender systems [24, 23]. For example, coverage describes the proportion of items that the recommender system can recommend or the proportion of users for which the recommender system can recommend items; privacy means the systems should not disclose user's preferences to third parties without permission; diversity might be very important in some recommendation scenarios where similar items may not be as useful for different users (e.g., recommendation for a household with people having different tastes). A robust recommender system should keep stable in the presence of attacks or misinformation. Explainable recommender system could provide users with intuitive explanations of why certain items are recommended to them. Scalability is also a key factor when making decisions as different recommendation scenarios may have different levels of tolerance on the computational overhead.

5 Conclusion

This chapter was structured around the techniques, applications, and evaluations of recommender systems. We introduced non-personalized methods, neighborhood method, factorization-based approaches, as well as recent deep-learning-based methods. These methods are applicable for a wide spectrum of recommendation tasks. We also discussed several specific recommendation applications and concerns of designing industrial-scale recommender systems. Moreover, we introduced a set of evaluation metrics that can be used for evaluating recommender systems.

The values and contributions of recommender systems cannot be overestimated. The development and advancement in the field are inspiring and enlightening. We hope that the panorama of recommender systems provided in this chapter can help

researchers and practitioners to get a deep understanding toward recommender systems.

References

1. Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. WWW.
2. Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 8 (2009): 30–37.
3. Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. KDD. ACM, New York, NY, USA, 426–434.
4. Ma, Hao, et al. "Recommender systems with social regularization." WSDM. ACM, 2011.
5. Mnih, Andriy, and Ruslan R. Salakhutdinov. "Probabilistic matrix factorization." Advances in Neural Information Processing Systems. 2008.
6. Rendle, Steffen, et al. "BPR: Bayesian personalized ranking from implicit feedback." UAI. AUAI Press, 2009.
7. Hsieh, Cheng-Kang, et al. "Collaborative metric learning." WWW, 2017.
8. Quadrana, Massimo, Paolo Cremonesi, and Dietmar Jannach. "Sequence-aware recommender systems." ACM Computing Surveys (CSUR) 51.4 (2018): 66.
9. Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized ranking metric embedding for next new POI recommendation. IJCAI.
10. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT Press, 2016.
11. Zhang, Shuai, et al. "Deep learning based recommender system: A survey and new perspectives." ACM Computing Surveys (CSUR) 52.1 (2019): 5.
12. He, Xiangnan, et al. "Neural collaborative filtering." WWW, 2017.
13. Sedhain, Suvash, et al. "Autorec: Autoencoders meet collaborative filtering." WWW. ACM, 2015.
14. Wu, Yao, et al. "Collaborative denoising auto-encoders for top-n recommender systems." WSDM. ACM, 2016.
15. Cheng, Heng-Tze, et al. "Wide & deep learning for recommender systems." Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM, 2016.
16. Guo, Huifeng, et al. "DeepFM: a factorization-machine based neural network for CTR prediction." arXiv preprint arXiv:1703.04247 (2017).
17. He, Kaiming, et al. "Deep residual learning for image recognition." CVPR. 2016.
18. Zhang, Shuai, et al. "Next Item Recommendation with Self-Attentive Metric Learning." AAAI Conference. Vol. 9. 2019.
19. Ying, Rex, et al. "Graph convolutional neural networks for web-scale recommender systems." Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 2018.
20. Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).
21. Zheng, Guanjie, et al. "DRN: A deep reinforcement learning framework for news recommendation." WWW, 2018.
22. Covington, Paul, Jay Adams, and Emre Sargin. "Deep neural networks for YouTube recommendations." RecSys. ACM, 2016.
23. Ricci, Francesco, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook." Recommender Systems Handbook. Springer, Boston, MA, 2011. 1–35.
24. Ge, Mouzhi, Carla Delgado-Battenfeld, and Dietmar Jannach. "Beyond accuracy: evaluating recommender systems by coverage and serendipity." RecSys. ACM, 2010.

25. Pazzani, Michael J., and Daniel Billsus. “Content-based recommendation systems.” In The adaptive web, pp. 325–341. Springer, Berlin, Heidelberg, 2007.
26. Burke, R., 2002. Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), pp.331–370.
27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention is all you need. arXiv preprint arXiv:1706.03762.
28. Barkan, O., Koenigstein, N., Yogev, E. and Katz, O., 2019, September. CB2CF: a neural multiview content-to-collaborative filtering model for completely cold item recommendations. RecSys (pp. 228–236).
29. Malkiel, I., Barkan, O., Caciularu, A., Razin, N., Katz, O. and Koenigstein, N., 2020. RecoBERT: A Catalog Language Model for Text-Based Recommendations. arXiv preprint arXiv:2009.13292.
30. Linden, Greg, Brent Smith, and Jeremy York. “Amazon.com recommendations: Item-to-item collaborative filtering.” IEEE Internet Computing 7, no. 1 (2003): 76–80.
31. Bennett, J. and Lanning, S., 2007, August. The Netflix Prize. In Proceedings of KDD Cup and Workshop (Vol. 2007, p. 35).
32. Koenigstein, N., Nice, N., Paquet, U. and Schleyen, N., 2012, September. The Xbox recommender system. In RecSys (pp. 281–284).
33. Barkan, O. and Koenigstein, N., 2016, September. Item2vec: neural item embedding for collaborative filtering. In 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP) (pp. 1–6). IEEE.
34. Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
35. Barkan, O., Fuchs, Y., Caciularu, A. and Koenigstein, N., 2020, September. Explainable recommendations via attentive multi-persona collaborative filtering. In RecSys (pp. 468–473).
36. Zhang, Yongfeng, and Xu Chen. “Explainable Recommendation: A Survey and New Perspectives.” Foundations and TrendsÁő in Information Retrieval 14, no. 1 (2020): 1–101.
37. Kang, W.C., Fang, C., Wang, Z. and McAuley, J., 2017, November. Visually-aware fashion recommendation and design with generative image models. ICDM (pp. 207–216). IEEE.
38. Phanich, M., Pholkul, P. and Phimoltares, S., 2010, April. Food recommendation system using clustering analysis for diabetic patients. In 2010 International Conference on Information Science and Applications (pp. 1–8). IEEE.
39. Goldberg, D., Nichols, D., Oki, B.M. and Terry, D., 1992. Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), pp. 61–70.
40. Vinh Tran, L., Tay, Y., Zhang, S., Cong, G. and Li, X., 2020, January. HyperML: a boosting metric learning approach in hyperbolic space for recommender systems. In Proceedings of the 13th International Conference on Web Search and Data Mining (pp. 609–617).
41. Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Activity Recognition

Jindong Wang, Yiqiang Chen, and Chunyu Hu

1 Introduction

The goal of ubiquitous computing is to provide easy-to-access computing and more natural human-computer interaction in anywhere and anytime. In ubiquitous computing, the key component is to sense the human activities before making further suggestions or recommendations. Human activity recognition (HAR) is of great importance in people's daily life, since it is able to learn profound high-level knowledge about human activity from raw sensor inputs. Through the years, there have been tremendous HAR applications in people's daily life, such as indoor localization (Xu et al., 2016), sleep state detection (Zhao et al., 2017), smart home sensing (Wen et al., 2016; Vepakomma et al., 2015), video surveillance (Qin et al., 2015), gait analysis (Hammerla et al., 2016), and gesture recognition (Kim and Toomajian, 2016).

HAR is a rather hot research topic in ubiquitous computing. It has close relationship to human computing interaction (HCI), user interface design, and crowd-sourcing. From this perspective, HAR can be treated as the bridge between low-level human activities and more high-level computations. In this chapter, our focus will be on sensor-based activity recognition (Wang et al., 2019), which recognizes activities using embedded sensors rather than those based on camera or videos.

From the machine learning perspective, HAR is a specific classification or regression problem. Figure 1 shows an overview of the HAR process. There are

J. Wang (✉)
Microsoft Research Asia, Beijing, China
e-mail: jindong.wang@microsoft.com

Y. Chen · C. Hu
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
e-mail: yqchen@ict.ac.cn; huchunyu@ict.ac.cn

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_29

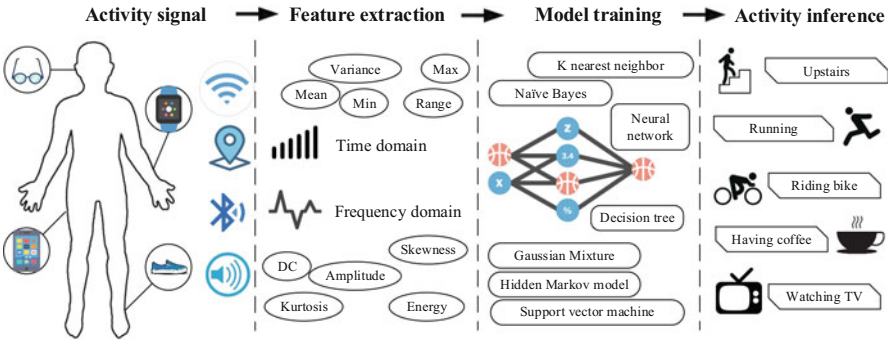


Fig. 1 The process of human activity recognition. Source: (Wang et al., 2019)

four main components: Activity signal sensing, feature extraction, model training, and activity inference. These four components are also the standard procedures of a machine learning or pattern recognition problem (Bishop, 2006). Therefore, HAR can be implemented by taking advantages of the machine learning algorithms. For example, a very early work on HAR was presented by (Bao and Intille, 2004), where authors used five small biaxial accelerometers worn by 20 subjects to recognize human activities. They extracted several features including mean, energy, frequency-mean and energy, and correlations of activities. After that, they tested several machine learning classifiers among which the decision tree classifier showed the best performance.

Conventional approaches have made tremendous progress on HAR by adopting machine learning algorithms such as decision tree, support vector machine, naive Bayes, and hidden Markov models (Lara and Labrador, 2012). In recent years, with the development of deep learning (LeCun et al., 2015) in learning representative features and unprecedented performance, there are a lot of work that adopt deep learning models for more accurate HAR tasks (Plötz et al., 2011; Lane et al., 2015; Alsheikh et al., 2016b). Compared to traditional machine learning that relies heavily on human-crafted feature design, deep learning is able to automatically learn high-level and meaningful features and thus can achieve better performance. Moreover, deep learning is more suitable for big data and online/incremental learning (Wang et al., 2019).

In this chapter, we will introduce the basics of HAR. In next sections, we first give a formal definition of HAR. Then, more detailed descriptions of the components in Fig. 1 will be presented. After that, we introduce the deep learning based HAR in recent years. Section 3 will introduce some hot research topics in HAR and representative works, and Sect. 5 provides some grand challenges in the future. Finally, Sect. 6 concludes this chapter.

2 The Procedure of Activity Recognition

2.1 Problem Formulation

HAR aims to understand human behaviors which enable the computing systems to proactively assist users based on their requirement (Bulling et al., 2014). Formally speaking, suppose a user is performing some kinds of activities belonging to a predefined activity set A:

$$A = \{A_i\}_{i=1}^{m} \tag{1}$$

where m denotes the number of activity types. There is a sequence of sensor reading that captures the activity information

$$\mathbf{s} = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_t, \cdots \mathbf{d}_n\} \tag{2}$$

where $\mathbf{d}_t$ denotes the sensor reading at time t.

We need to build a model $\mathcal{F}$ to predict the activity sequence based on sensor reading $\mathbf{s}$

$$\hat{A} = \{\hat{A}_j\}_{j=1}^{n} = \mathcal{F}(\mathbf{s}), \quad \hat{A}_j \in A \tag{3}$$

while the true activity sequence (ground truth) is denoted as

$$A^* = \{A_j^*\}_{j=1}^{n}, \quad A_j^* \in A \tag{4}$$

where n denotes the length of sequence and $n \geq m$.

The goal of HAR is to learn the model $\mathcal{F}$ by minimizing the discrepancy between predicted activity $\hat{A}$ and the ground truth activity A^*. Typically, a positive loss function $\mathcal{L}(\mathcal{F}(\mathbf{s}), A^*)$ is constructed to reflect their discrepancy. $\mathcal{F}$ usually does not directly take $\mathbf{s}$ as input, and it usually assumes that there is a projection function Φ that projects the sensor reading data $\mathbf{d_i} \in \mathbf{s}$ to a d-dimensional feature vector $\Phi(\mathbf{d}_i) \in \mathbb{R}^d$. To that end, the goal turns into minimizing the loss function $\mathcal{L}(\mathcal{F}(\Phi(\mathbf{d}_i)), A^*)$.

2.2 Sensor Inputs

In sensor-based HAR, there are multiple kinds of sensors that can be adopted to collect user data. Although some HAR approaches can be generalized to all sensor modalities, most of them are only specific to certain types. According to (Chavarriaga et al., 2013), we mainly classify those modalities into three aspects: *body-worn sensors*, *object sensors*, and *ambient sensors*.

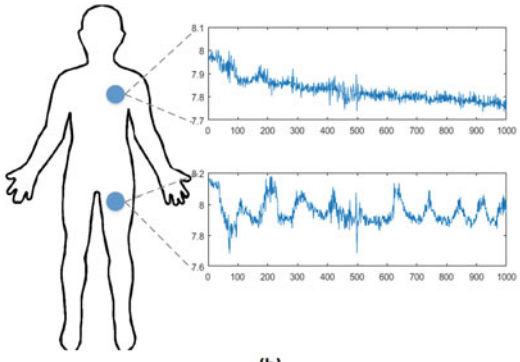
Body-worn sensors are sensors worn by the user to describe the body movements. This kind of sensors is the most popular and affordable ones in HAR. In today's era filled with smart devices, a lot of devices such as smartphone, smart watch, wristband, smart glasses can all be used as sensors. The key components in these devices are accelerometers and gyroscopes, which are important to describe users' movements. Object sensors are those attached to objects to capture object movements. A popular technology in this category is Radio Frequency IDentification (RFID), which is pretty popular in measuring the interactions with the objects. On the other hand, sensors attached to a certain object can also be regarded as an object sensor. The last category is the ambient sensor, which are applied in environments to reflect user interaction with the environment. This category lays more focus on the environment, such as sound sensors, Wi-Fi, Bluetooth, temperature sensors, etc.

Human activities are happening with a range of time period. This is its nature. Therefore, all of these sensors are recording signals in a form of time series data. For instance, the data collected by an accelerometer is a multi-dimensional time series data. Assuming the accelerometer has 3 axis, then, the activity data will have a $L \times (1 + 3 + 1)$ shape, where the column `Timestamp` denotes the timestamps, columns `X-Y-Z` denote the axis, and `Label` denotes the activity categories (often encoded in numbers). An example can be found in Fig. 2, where the sensors are placed at different positions of the subject.

For most situations, the sensor readings are with noise and have to go through a filtering process. A common practice is to use the high-pass or low-pass filter, where a threshold is set such that any value higher or lower than that threshold has to be omitted. In order to get more smooth movement curve, some researchers used Karman filter to further filter the signals. In this way, the preprocess of the sensor data can be regarded as a signal processing problem, where the noise and outliers have to be removed for better performance later (Bao and Intille, 2004).

Timestamp	X	Y	Z	Label
1568533339	8.1305	1.0349	5.4217	1
1568533340	8.1305	1.0202	5.3843	1
1568533341	8.1604	1.0201	5.3622	1
1568533342	8.1603	1.0052	5.377	2
1568533343	8.1605	1.0275	5.3473	2
1568533344	8.1454	1.0201	5.3919	2
1568533345	8.1454	1.0495	5.4518	2
1568533346	8.1376	0.99744	5.5111	3
1568533347	8.1307	1.0646	5.3921	3
1568533348	8.1155	1.0202	5.4215	3
1568533349	8.1605	1.0496	5.3773	3
1568533350	8.1306	1.0204	5.3396	3

(a)



(b)

Fig. 2 Sensor inputs and readings of a subject. (**a**) Sensor inputs. (**b**) Sensor readings

2.3 Feature Engineering

After sensor inputs preprocessing, another important step is feature engineering. Before we get into feature engineering, it is necessary to know that this the raw inputs can also be directly used as features for HAR, only with worse performance since the raw inputs fail to fully represent the signal features. In a machine learning point of view, the features are necessary and important for the classification problem since they are critical and representative of the characteristics of the inputs.

Before the emerge of deep learning, traditional machine learning algorithms have to rely on hand-crafted feature engineering. That is, the features are designed and extracted according to human knowledge (Lara and Labrador, 2012; Wang et al., 2019). According to existing work, the features are mainly extracted in two categories: time domain and frequency domain. Several common features in these two domains are listed in Table 1.

After feature extraction, there are often a feature selection and dimensionality reduction process. The nature of this step is to use fewer features to represent the most information so as to reduce the computation burden and even eliminate the curse of dimensionality (Bishop, 2006). Generally speaking, features can be selected according to their relative importance judged by human experience. A popular way of feature selection is to use decision trees or forest models to rand all the features. Dimensionality reduction can also reduce the feature numbers, in an implicit way. The dimensionality reduction techniques such as Principle Component Analysis (PCA), Kernel PCA, Linear Discriminant Analysis (LDA) (Bishop, 2006) can all be used.

These days, with the development of deep learning, features can be automatically extracted in a neural network without human knowledge. In a deep neural network, features can be extracted in different layers and controlled by the user, leading to

Table 1 Several common features extracted in time and frequency domain

ID	Feature	Description
1	Mean	Average value of samples in window
2	STD	Standard deviation
3	Minimum	Minimum
4	Maximum	Maximum
5	Mode	The value with the largest frequency
6	Range	Maximum minus minimum
7	Mean crossing rate	Rate of times signal crossing mean value
8	DC	Direct component
9–13	Spectrum peak position	First 5 peaks after FFT
14–18	Frequency	Frequencies corresponding to 5 peaks
19	Energy	Square of norm
20–23	Four shape features	Mean, STD, skewness, kurtosis
24–27	Four amplitude features	Mean, STD, skewness, kurtosis

different kinds of features that are learned through the network (LeCun et al., 2015). In this way, we can think of deep representation learning as a more generalized case of traditional hand-crafted feature extraction. Feature selection and dimensionality reduction can also be done within the network. And experiments have shown that deep features are more representative than traditional features and lead to better performance (Plötz et al., 2011).

After data cleaning and normalization, features can be extracted. It is noteworthy that when performing feature engineering, the *sliding window* strategy is often used. The inputs should be cut into individual inputs according to the sampling rate. For instance, when the sensor has a sampling frequency of 50 Hz, we often adopt a sliding window size of 2 s, with an overlap of 50%. This should ensure that the repetitive features can be extracted.

2.4 Model

Before deep learning, researchers used traditional machine learning algorithms for accurate HAR. There are a lot of algorithms available, including support vector machines (SVM), hidden Markov models (HMM), decision trees (DT), logistic regression (LR), and random forest (RF). Existing works used these algorithms to build accurate as well as personalized models (Bao and Intille, 2004; Kwapisz et al., 2011; Lara and Labrador, 2012).

Most of the HAR problem can be treated as a classification problem except for Wi-Fi-based or Bluetooth-based indoor localization, which is a regression problem. In these algorithms, extracted features are the inputs. Then, the algorithm will train a model accordingly.

Deep learning makes it possible to perform end-to-end HAR. Table 2 lists several popular models used in deep learning based HAR (Wang et al., 2019). In fact, any deep networks can be used for HAR.

A natural question arises: *which model is the best for HAR?* (Hammerla et al., 2016) did an early work by investigating the performance of DNN, CNN, and RNN through 4000 experiments on some public HAR datasets. We combine their work and our explorations to draw some conclusions: RNN and LSTM are recommended

Table 2 Deep learning models for HAR tasks

Model	Description
MLP	Deep fully connected network, artificial neural network with deep layers
CNN	Convolutional neural network, multiple convolution operations for feature extraction
RNN	Recurrent neural network, network with time correlations and LSTM
DBN/RBM	Deep belief network and restricted Boltzmann machine
SAE	Stacked autoencoder, feature learning by decoding-encoding autoencoder
Hybrid	Combination of some deep models

to recognize short activities that have natural order while CNN is better at inferring long-term repetitive activities (Hammerla et al., 2016). The reason is that RNN could make use of the time-order relationship between sensor readings, and CNN is more capable of learning deep features contained in recursive patterns. For multi-modal signals, it is better to use CNN since the features can be integrated through multi-channel convolutions (Zeng et al., 2014; Ha et al., 2015). While adapting CNN, data-driven approaches are better than model-driven approaches as the inner properties of the activity signal can be exploited better when the input data are transformed into the virtual image. Multiple convolutions and poolings also help CNN perform better. RBM and autoencoders are usually pre-trained before being fine-tuned. Multi-layer RBM or SAE is preferred for more accurate recognition.

Technically there is no model which outperforms all the others in all situations, so it is recommended to choose models based on the scenarios. Moreover, the hybrid models tend to perform better than single models (DeepConvLSTM in OPPORTUNITY 1 and Skoda). For a single model, CNN with shifted inputs (Fourier transform) generates better results compared to shifted kernels.

2.5 Evaluation

The evaluation of HAR is based on statistical results comparing the ground truth and the results given by models. The *accuracy* is the most popular metric for evaluating HAR models. It can be defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{5}$$

where TP is the true positive, indicating the number of positive instances that are classified as positive; TN is true negative, which represents the number of negative samples that are classified as negative; FP is false positive, meaning the number of negative instances that are classified as positive; FN is false negative, which stands for the number of positive instances that are classified as negative.

However, in a multi-class classification problem, accuracy alone may not be enough since there are often imbalanced classification situations. Therefore, the *precision, recall*, and F measures are used.

The precision (P) and recall (R) can be calculated as follows:

$$P = \frac{TP}{TP + FP} \tag{6}$$

$$R = \frac{TP}{TP + FN} \tag{7}$$

The *F-measure* denotes the combination of precision and recall:

$$F\text{-}measure = \frac{2 \cdot P \cdot R}{P + R} \tag{8}$$

As for a regression problem, there are two types of evaluations that can be used: Mean Average Error (MAE) and Mean Squared Error (MSE). Denote n the number of test samples, y_i and f_i are the true and predicted values, respectively, then, the evaluations can be calculated as follows:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |f_i - y_i| \tag{9}$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (f_i - y_i)^2 \tag{10}$$

When evaluating models, it is important to follow the standard protocols in machine learning to perform model selection (Bishop, 2006).

3 Hot Research Areas

In this section, we introduce several hot research areas for HAR. Despite the success of HAR, there are still existing challenges that are hot research areas these days. When the data distributions from different persons are different, we need transfer learning and domain adaptation to help build cross-domain models. Under the regulations of privacy preservation, we need federated learning to build models without leaking any user privacy. Finally, with the traditional models being static and fail to update fast online, we introduce incremental learning for fast update of the models. Note that these three areas are not currently well-explored and need more investigation in the future.

3.1 Transfer Learning Based HAR

HAR is a machine learning application, which requires a large amount of labeled data to train a powerful model. However, when there is not enough labeled data, the model is likely to perform poor. Assume a person is suffering from Small Vessel Disease (SVD) (Wardlaw et al., 2013), which is a severe brain disease heavily related to activities. However, we cannot equip his all body with sensors to acquire the labels since this will make his activities unnatural. We can only label the activities on certain body parts in reality. If the doctor wants to see his activity

information on the arm (we call it the *target domain*), which only contains sensor readings instead of labels, how to utilize the information on other parts (such as torso or leg, we call them the *source domains*) to help obtain the labels on the target domain? This is referred to as the *cross-domain activity recognition (CDAR).*

The problem of CDAR is extremely challenging. Firstly, we do not know which body part is the most similar to the target position since the sensor signals are not independent, but highly correlated because of the shared body structures and functions. If we use all the body parts as the source domain, there is likely to be *negative* transfer (Pan and Yang, 2010) because some body parts may be dissimilar. Secondly, we only have the raw activity data on the target domain without the actual activity labels, making it infeasible to measure the similarities between different body positions. Thirdly, even when we know the similar body parts to the target domain, it is still difficult to build a good machine learning model using both the source and the target domains. The reason is that signals from different domains are following different distributions, which means there are distribution discrepancies between them. However, traditional machine learning models are built by assuming that all signals follow the same distribution. Fourthly, when it comes to *multiple* persons, the sensor readings are more different compared to different body parts on one person. This makes the problem more challenging.

The problem of CDAR consists of two parts: determining the most similar source domain to the target domain, and perform transfer learning accordingly. Several transfer learning methods have been proposed (Cook et al., 2013). The key is to learn and reduce the distribution divergence (distance) between two domains. With the distance, we can perform source domain selection as well as knowledge transfer. Based on this principle, existing methods can be summarized into two categories: exploiting the correlations between features (Blitzer et al., 2006; Kouw et al., 2016), or transforming both the source and the target domains into a new shared feature space (Pan et al., 2011; Gong et al., 2012; Wang et al., 2017). Among existing work on transfer learning based HAR (Cook et al., 2013), Zhao et al. proposed a transfer learning method TransEMDT (Zhao et al., 2011) using decision trees, but it ignored the intra-class similarity within classes. (Khan and Roy, 2017) proposed the TransAct framework, which is a boosting-based method and ignores the feature transformation procedure. Thus it is not feasible in most activity cases. Feuz et al. (Feuz and Cook, 2017) proposed a heterogeneous transfer learning method for HAR, but it only learns a global domain shift. A more recent work of Wang et al. (Wang et al., 2018b) performs activity recognition using deep transfer learning. Recently, the work of (Wang et al., 2018a; Chen et al., 2019a) proposed the notion of *Stratified Transfer Learning*, which focuses on the local distance between domains and can perform accurate source domain selection and activity transfer (Fig. 3).

Transfer learning based HAR has many potential applications:

1. *Activity recognition.* The results of activity recognition can be different according to different *devices*, *users*, and wearing *positions*. Transfer learning makes it possible to perform cross-device/user/position activity recognition with high

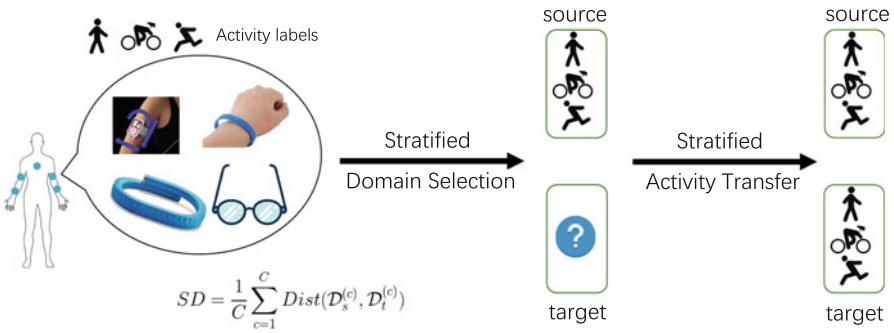


Fig. 3 The stratified transfer learning framework proposed by (Wang et al., 2018a)

accuracy. In case cross-domain learning is needed, finding and measuring the similarity between the device/user/position is critical.

2. *Localization.* In Wi-Fi localization, the Wi-Fi signal changes with the *time*, *sensor*, and *environment*, causing the distributions different. So it is necessary to perform cross-domain localization. When applying STL to this situation, it is also important to capture the similarity of signals according to time/sensor/environment.
3. *Gesture recognition.* For gesture recognition, due to differences in hand structure and moving patterns, the model cannot generalize well. In this case, transfer learning can also be a good option. Meanwhile, special attention needs to be paid to the divergence between the different characteristic of the subjects.
4. *Other context-related applications.* Other applications include smart home sensing, intelligent city planning, healthcare, and human-computer interaction. They are also context-related applications. Most of the models built for pervasive computing are only *specific* to certain contexts. Transfer learning makes it possible to transfer the knowledge between related contexts, of which transfer learning can achieve the best performance. But when recognizing high-level contexts such as *Coffee Time*, it is rather important to consider the relationship between different contexts in order to utilize their similarities. The research on this area is still on the go.

3.2 Federated Learning Based HAR

In healthcare applications, machine learning models are often trained on sufficient user data to track health status. Traditional machine learning approaches such as Support Vector Machines (SVM), Decision Tree (DT), and Hidden Markov Models (HMM) are adopted in many healthcare applications (Wang et al., 2019). The recent success of deep learning achieves satisfactory performances by training on larger sizes of user data. Representative networks include Convolutional Neural

Networks (CNN), Recurrent Neural Networks (RNN), and Autoencoders (Wang et al., 2019).

Unfortunately, there is one critical challenge in today's wearable healthcare. First of all, in real life, data often exists in the form of isolated islands. Although there are plenty of data in different organizations, institutes, and subjects, it is not possible to share them due to privacy and security concerns. When the same user uses different products from two companies, his data stored in two clouds cannot be exchanged. This makes it hard to train powerful models using these valuable data. Additionally, recently, China, the United States, and the European Union enforced the protection of user data via different regularizations. Hence, the acquisition of massive user data is not possible in real applications.

A comprehensive survey on federated learning is in (Yang et al., 2019). Federated machine learning was firstly proposed by Google (Konečnỳ et al., 2016), where they trained machine learning models based on distributed mobile phones all over the world. The key idea is to protect user data during the process. Since then, other researchers started to focus on privacy-preserving machine learning (Bonawitz et al., 2017; Geyer et al., 2017), federated multi-task learning (Smith et al., 2017), as well as personalized federated learning (Chen et al., 2018). Federated learning has the ability to resolve the data islanding problems by privacy-preserving model training in the network.

According to (Yang et al., 2019), federated learning can mainly be classified into three types: (1) horizontal federated learning, where organizations share partial features; (2) vertical federated learning, where organizations share partial samples; and (3) federated transfer learning, where neither samples nor features have much in common. FedHealth belongs to federated transfer learning category. It is the first of its kind tailored for wearable healthcare applications.

Recently, Chen et al. proposed a new wearable healthcare framework called *FedHealth* based on federated learning. FedHealth aims to achieve accurate personal healthcare through federated transfer learning without compromising privacy security. Figure 4 gives an overview of the framework. Without loss of generality, we assume there are 3 users (organizations) and 1 server, which can be extended to the more general case. The framework mainly consists of four procedures. First of all, the cloud model on the server end is train based on public datasets. Then, the cloud model is distributed to all users where each of them can train their own model on their data. Subsequently, the user model can be uploaded to the cloud to help train a new cloud model. Note that this step does not share any user data or information but the encrypted model parameters. Finally, each user can train personalized models by integrating the cloud model and its previous model and data for personalization. In this step, since there is large distribution divergence between cloud and user model, transfer learning is performed to make the model more tailored to the user (right part in Fig. 4). It is noteworthy that all the parameter sharing processes does not involve any leakage of user data. Instead, they are finished through homomorphic encryption (Rivest et al., 1978).

The federated learning paradigm is the main computing model for the whole FedHealth framework. It deals with model building and parameter sharing during

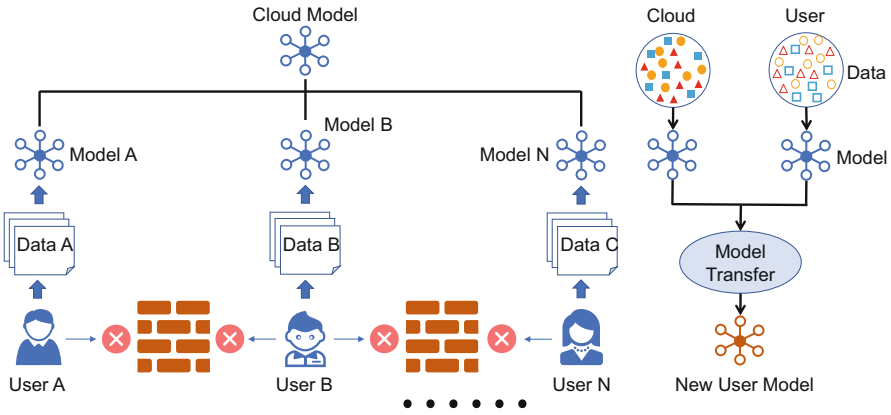


Fig. 4 The FedHealth framework proposed by (Chen et al., 2019b)

the entire process. After the server model is learned, it can be directly applied to the user. This is just what traditional healthcare applications do for model learning. It is obvious that the samples in the server are having highly different probability distribution with the data generated by each user. Therefore, the common model fails in personalization. Additionally, user models cannot easily be updated continuously due to the privacy security issue.

FedHealth is a general framework for wearable healthcare. This paper provides a specific implementation and evaluation of this idea. It is adaptable to several healthcare applications. In this section, we discuss its potential to be extended and deployed to other situations with possible solutions.

1. FedHealth with incremental learning. Incremental learning (Rebuffi et al., 2017) has the ability to update the model with the gradually changing time, environment, and users. In contrast to transfer learning that focuses on model adaptation, incremental learning makes it possible to update the model in real-time without much computation.
2. FedHealth as the standard for wearable healthcare in the future. FedHealth provides such a platform where all the companies can safely share data and train models. In the future, we expect that FedHealth be implemented with blockchain technology (Zheng et al., 2018) where user data can be more securely stored and protected. We hope that FedHealth can become the standard for wearable healthcare.
3. FedHealth to be applied in more applications. This work mainly focuses on the possibility of federated transfer learning in healthcare via activity recognition. In real situations, FedHealth can be deployed at large-scale to more healthcare applications such as elderly care, fall detection, cognitive disease detection, etc. We hope that through FedHealth, federated learning can become federated computing which can become a new computing model in the future.

3.3 Incremental Learning Based HAR

Commonly, activity recognition models are designed for a static environment, which is not applicable to a real scenario. These models are trained with data collected offline. Once the training phase is completed, the model is fixed throughout the activity recognition process. However, fixed recognition models are difficult to meet the need in practical usage. Many factors, including changes in user, activity class, and sensor, will lead to the failure of a fixed model. Incremental learning is an effective way to handle these challenges.

In most cases, HAR model is built with data collected from predefined users. However, it is not able to fit for a specified subject, which is not included in the initial training set. This can be attributed to that data collected from different users conform to different distributions. To handle changes in the distribution of input data, (Saffari et al., 2009; Elgawi, 2008; Denil et al., 2013; Lakshminarayanan et al., 2014) proposed four variants of online random forest algorithms. In (Sztyler and Stuckenschmidt, 2017), ORF-Saffari is combined with active learning to construct a cross-subject activity recognition model. Experimental results show that it is efficient in building personalized activity recognition classifiers. These methods focus on the dynamic changes of data distribution. They aim to refine the function $f : \mathbb{R}^K \rightarrow Y$ with new data from previously known features.

On the other hand, people is able to learn new activities with time pass by. When a new kind of activity is performed, devices with preinstalled activity recognition models may fail to recognize new activities. To recognize new activities without retraining from scratch, (Zhao et al., 2014) proposed the CIELM class incremental learning method. With CIELM, new activities can be recognized dynamically. In (Hu et al., 2018), Hu et al. designed a separating axis theorem based splitting strategy to insert internal nodes. Combined with splitting leaf nodes, their incremental learning method can match the performance of random forest. Ristin et al. proposed two variants of the random forest model to avoid retraining from scratch (Ristin et al., 2016) when handling the class incremental learning problem. It can extend random forest initially trained with just 10 classes to 1000 classes with an acceptable loss of accuracy. These work attempts to learn the function $f : \mathbb{R}^K \rightarrow Y'$ by focusing on handling changes in the output space.

Traditional sensor-based activity recognition methods train fixed classification models with labeled data collected offline. Such models are unable to adapt to dynamic changes in real applications. With the emerging of new wearable devices, more diverse sensors can be used to improve the performance of activity recognition. However, it is difficult to integrate a new sensor into a pre-trained activity recognition model. The emergence of new sensors will lead to a corresponding increase in the feature dimensionality of the input data, which may result in the failure of a pre-trained activity recognition model. In (Hu et al., 2019), Hu et al. propose a novel feature incremental learning method, namely Feature Incremental Random Forest (Fig. 5). It is able to adapt an existing activity recognition model

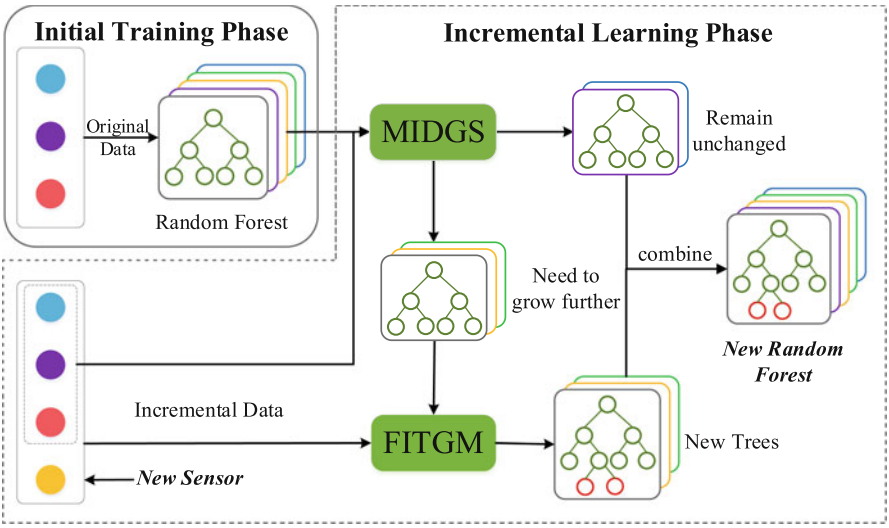


Fig. 5 The feature incremental random forest proposed by (Hu et al., 2019)

to the new emerging sensors. Both (Rey and Lukowicz, 2017) and (Bannach et al., 2011) proposed methods that can incorporate new sensors into an existing activity recognition system in an unsupervised manner. In (Bannach et al., 2011), the clustering membership of the higher dimensional data and heuristics were utilized to label newly arriving data. In (Rey and Lukowicz, 2017), similarity and transduction were used to label such new data. Similarly, Gregory et al. (Amis and Carpenter, 2010) proposed a self-supervised learning method—ARTMAP. It is able to learn features from unlabeled patterns without losing knowledge previously acquired from labeled patterns.

Incremental learning provide an alternative way to handle with changes in an opening environment, which is of great significance in real applications. It is able to provide more intelligent, personalized, and convenient services to users. There are still some challenges to be solved in the future. How to design a more smart incremental activity recognition model? No matter what the change is, it is able to capture the difference as soon as possible.

4 Datasets

Table 3 listed some popular HAR datasets that are actively used in previous works.

Table 3 Public HAR datasets (*A* accelerometer, *G* gyroscope, *M* magnetometer, *O* object sensor, *AM* ambient sensor, *ECG* electrocardiograph)

ID	Dataset	Type	#Subject	S. Rate	#Activity	#Sample	Sensor	Reference
D01	OPPORTUNITY	ADL	4	32 Hz	16	701,366	A, G, M, O, AM	Ordóñez and Roggen, 2016
D02	Skoda checkpoint	Factory	1	96 Hz	10	22,000	A	Plötz et al., 2011
D03	UCI smartphone	ADL	30	50 Hz	6	10,299	A, G	Almaslukh et al., 2017
D04	PAMAP2	ADL	9	100 Hz	18	2,844,868	A, G, M	Zheng et al., 2014
D05	USC-HAD	ADL	14	100 Hz	12	2,520,000	A, G	Jiang and Yin, 2015
D06	WISDM	ADL	29	20 Hz	6	1,098,207	A	Alsheikh et al., 2016a
D07	DSADS	ADL	8	25 Hz	19	1,140,000	A, G, M	Zhang et al., 2015
D08	Ambient kitchen	Food preparation	20	40 Hz	2	55,000	O	Plötz et al., 2011
D09	Darmstadt daily routines	ADL	1	100 Hz	35	24,000	A	Plötz et al., 2011
D10	Actitracker	ADL	36	20 Hz	6	2,980,765	A	Zeng et al., 2014
D11	SHO	ADL	10	50 Hz	7	630,000	A, G, M	Jiang and Yin, 2015
D12	BIDMC	Heart failure	15	125 Hz	2	>20,000	ECG	Zheng et al., 2014
D13	MHEALTH	ADL	10	50 Hz	12	16,740	A, C, G	Ha and Choi, 2016
D14	Daphnet gait	Gait	10	64 Hz	2	1,917,887	A	Hammerla et al., 2016
D15	ActiveMiles	ADL	10	50–200 Hz	7	4,390,726	A	Ravì et al., 2017
D16	HASC	ADL	1	200 Hz	13	NA	A	Hayashi et al., 2015
D17	PAF	PAF	48	128 Hz	2	230,400	EEG	Pourbabaee et al., 2017
D18	ActRecTut	Gesture	2	32 Hz	12	102,613	A, G	Yang et al., 2015
D19	Heterogeneous	ADL	9	100–200 Hz	6	43,930,257	A, G	Yao et al., 2017

5 Challenges

Despite the progress in previous work, there are still challenges for deep learning based HAR. In this section, we present those challenges and propose some feasible solutions.

5.1 Online and Mobile Deep Activity Recognition

Two critical issues are related to deep HAR: online deployment and mobile application. Although some existing work adopted deep HAR on smartphone (Lane et al., 2015) and watch (Bhattacharya and Lane, 2016), they are still far from online and mobile deployment. Because the model is often trained offline on some remote server and the mobile device only utilizes a trained model. This approach is neither real-time nor friendly to incremental learning. There are two approaches to tackle this problem: *reducing the communication cost between mobile and server*, and *enhancing computing ability of the mobile devices*.

5.2 More Accurate Unsupervised Activity Recognition

The performance of deep learning still relies heavily on labeled samples. Acquiring sufficient activity labels is expensive and time-consuming. Thus, *unsupervised* activity recognition is urgent.

- *Take advantage of the crowd.* The latest research indicates that exploiting the knowledge from the crowd will facilitate the task (Prelec et al., 2017). Crowdsourcing takes advantage of the crowd to annotate the unlabeled activities. Other than acquiring labels passively, researchers could also develop more elaborate, privacy-concerned way to collect useful labels.
- *Deep transfer learning.* Transfer learning performs data annotation by leveraging labeled data from other auxiliary domains (Pan and Yang, 2010; Cook et al., 2013; Wang et al., 2017). There are many factors related to human activity, which can be exploited as auxiliary information using deep transfer learning. Problems such as sharing weights between networks, exploiting knowledge between activity related domains, and how to find more relevant domains are to be resolved.

5.3 Flexible Models to Recognize High-Level Activities

More complex high-level activities need to be recognized other than only simple daily activities. It is difficult to determine the hierarchical structure of high-level activities because they contain more semantic and context information. Existing methods often ignore the correlation between signals, thus they cannot obtain good results.

- *Hybrid sensor*. Elaborate information provided by the hybrid sensor is useful for recognizing fine-grained activities (Vepakomma et al., 2015). Special attention should be paid to the recognition of fine-grained activities by exploiting the collaboration of hybrid sensors.
- *Exploit context information*. Context is any information that can be used to characterize the situation of an entity (Abowd et al., 1999). Context information such as Wi-Fi, Bluetooth, and GPS can be used to infer more environmental knowledge about the activity. The exploitation of resourceful context information will greatly help to recognize user state as well as more specific activities.

5.4 Light-Weight Deep Models

Deep models often require lots of computing resources, which is not available for wearable devices. In addition, the models are often trained offline which cannot be executed in real-time. However, less complex models such as shallow NN and conventional PR methods could not achieve good performance. Therefore, it is necessary to develop light-weight deep models to perform HAR.

- *Combination of human-crafted and deep features*. Recent work indicated that human-crafted and deep features together could achieve better performance (Plötz et al., 2011). Some pre-knowledge about the activity will greatly contribute to more robust feature learning in deep models (Stewart and Ermon, 2017). Researchers should consider the possibility of applying two kinds of features to HAR with human experience and machine intelligence.
- *Collaboration of deep and shallow models.* Deep models have powerful learning abilities, while shallow models are more efficient. The collaboration of those two models has the potential to perform both accurate and light-weight HAR. Several issues such as how to share the parameters between deep and shallow models are to be addressed.

5.5 Non-invasive Activity Sensing

Traditional activity collection strategies need to be updated with more non-invasive approaches. Non-invasive approaches tend to collect information and infer activity without disturbing the subjects and requires more flexible computing resources.

- *Opportunistic activity sensing with deep learning.* Opportunistic sensing could dynamically harness the non-continuous activity signal to accomplish activity inference. In this scenario, back propagation of deep models should be well-designed.

5.6 Beyond Activity Recognition: Assessment and Assistant

Recognizing activities is often the initial step in many applications. For instance, some professional skill assessment is required in fitness exercises and smart home assistant plays an important role in healthcare services. There is some early work on climbing assessment (Khan et al., 2015). With the advancement of deep learning, more applications should be developed to be beyond just recognition.

6 Conclusions

In this chapter, we extensively introduce the activity recognition problem in ubiquitous computing area. We first formulate HAR into a machine learning problem. Then, we introduce the standard procedure to perform accurate HAR. Subsequently, we present two hot research areas of HAR: transfer learning aims at improving the model adaptability, while federated learning can help build models without compromising the security and privacy issues. Then, we introduce some grand challenges of HAR and propose some feasible solutions. HAR is extremely important to people's daily life. We hope that there can be more HAR research and applications in the future.

References

G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer, 1999.

B. Almaslukh, J. AlMuhtadi, and A. Artoli. An effective deep autoencoder approach for online smartphone-based human activity recognition. *International Journal of Computer Science and Network Security*, 17 (4): 160, 2017.

M. A. Alsheikh, A. Selim, D. Niyato, L. Doyle, S. Lin, and H.-P. Tan. Deep activity recognition models with triaxial accelerometers. *AAAI workshop*, 2016a.

M. A. Alsheikh, A. Selim, D. Niyato, L. Doyle, S. Lin, and H.-P. Tan. Deep activity recognition models with triaxial accelerometers. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016b.

G. P. Amis and G. A. Carpenter. Self-supervised ARTMAP. *Neural Netw.*, 23 (2): 265–282, 2010. ISSN 08936080. https://doi.org/10.1016/j.neunet.2009.07.026.

D. Bannach, B. Sick, and P. Lukowicz. Automatic adaptation of mobile activity recognition systems to new sensors. In *Proc. Ubicomp'11*, 2011.

L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *International conference on pervasive computing*, pages 1–17. Springer, 2004.

S. Bhattacharya and N. D. Lane. From smart to deep: Robust activity recognition on smartwatches using deep learning. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6. IEEE, 2016.

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *EMNLP*, pages 120–128, 2006.

K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017.

A. Bulling, U. Blanke, and B. Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46 (3): 33, 2014.

R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34 (15): 2033–2042, 2013.

F. Chen, Z. Dong, Z. Li, and X. He. Federated meta-learning for recommendation. *arXiv preprint arXiv:1802.07876*, 2018.

Y. Chen, J. Wang, M. Huang, and H. Yu. Cross-position activity recognition with stratified transfer learning. *Pervasive and Mobile Computing*, 57: 1–13, 2019a.

Y. Chen, J. Wang, C. Yu, W. Gao, and X. Qin. Fedhealth: A federated transfer learning framework for wearable healthcare. *arXiv preprint arXiv:1907.09173*, 2019b.

D. Cook, K. D. Feuz, and N. C. Krishnan. Transfer learning for activity recognition: A survey. *Knowledge and information systems*, 36 (3): 537–556, 2013.

M. Denil, D. Matheson, and D. Nando. Consistency of online random forests. In *Proc. ICML'13*, pages 1256–1264, 2013.

O. H. Elgawi. Online random forests based on CorrFS and CorrBE. In *CVPRW'08*, pages 1–7. IEEE, 2008.

K. D. Feuz and D. J. Cook. Collegial activity learning between heterogeneous sensors. *Knowledge and Information Systems*, pages 1–28, 2017.

R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pages 2066–2073, 2012.

S. Ha and S. Choi. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 381–388. IEEE, 2016.

S. Ha, J.-M. Yun, and S. Choi. Multi-modal convolutional neural networks for activity recognition. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3017–3022. IEEE, 2015.

N. Y. Hammerla, S. Halloran, and T. Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*, 2016.

T. Hayashi, M. Nishida, N. Kitaoka, and K. Takeda. Daily activity recognition based on DNN using environmental sound and acceleration signals. In *Signal Processing Conference (EUSIPCO), 23rd European*, pages 2306–2310, 2015.

C. Hu, Y. Chen, L. Hu, and X. Peng. A novel random forests based class incremental learning method for activity recognition. *Pattern Recognit.*, 78: 277–290, 2018. ISSN 00313203. https://doi.org/10.1016/j.patcog.2018.01.025.

C. Hu, Y. Chen, X. Peng, H. Yu, C. Gao, and L. Hu. A novel feature incremental learning method for sensor-based activity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 31 (6): 1038–1050, 2019.

W. Jiang and Z. Yin. Human activity recognition using wearable sensors by deep convolutional neural networks. In *MM*, pages 1307–1310. ACM, 2015.

A. Khan, S. Mellor, E. Berlin, R. Thompson, R. McNaney, P. Olivier, and T. Plötz. Beyond activity recognition: skill assessment from accelerometer data. In *UbiComp*, pages 1155–1166. ACM, 2015.

M. A. A. H. Khan and N. Roy. Transact: Transfer learning enabled activity recognition. In *PerCom Workshops*, pages 545–550. IEEE, 2017.

Y. Kim and B. Toomajian. Hand gesture recognition using micro-doppler signatures with convolutional neural network. *IEEE Access*, 4: 7125–7130, 2016.

J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

W. M. Kouw, L. J. van der Maaten, J. H. Krijthe, and M. Loog. Feature-level domain adaptation. *JMLR*, 17 (171): 1–32, 2016.

J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12 (2): 74–82, 2011.

B. Lakshminarayanan, D. M. Roy, and Y. W. Teh. Mondrian forests: Efficient online random forests. In *Adv. Neural. Inf. Process. Syst.*, pages 3140–3148. 2014.

N. D. Lane, P. Georgiev, and L. Qendro. DeepEar: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 283–294. ACM, 2015.

O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15 (3): 1192–1209, 2012.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521 (7553): 436, 2015.

F. J. Ordóñez and D. Roggen. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16 (1): 115, 2016.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22 (10): 1345–1359, 2010.

S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE TNN*, 22 (2): 199–210, 2011.

T. Plötz, N. Y. Hammerla, and P. L. Olivier. Feature learning for activity recognition in ubiquitous computing. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

B. Pourbabaee, M. J. Roshtkhari, and K. Khorasani. Deep convolution neural networks and learning ECG features for screening paroxysmal atrial fibrillation patients. *IEEE Trans. on Systems, Man, and Cybernetics*, 2017.

D. Prelec, H. S. Seung, and J. McCoy. A solution to the single-question crowd wisdom problem. *Nature*, 541 (7638): 532–535, 2017.

J. Qin, L. Liu, Z. Zhang, Y. Wang, and L. Shao. Compressive sequential learning for action similarity labeling. *IEEE Transactions on Image Processing*, 25 (2): 756–769, 2015.

D. Ravì, C. Wong, B. Lo, and G.-Z. Yang. A deep learning approach to on-node sensor data analytics for mobile or wearable devices. *IEEE journal of biomedical and health informatics*, 21 (1): 56–64, 2017.

S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

V. F. Rey and P. Lukowicz. Label propagation: An unsupervised similarity based method for integrating new sensors in activity recognition systems. *Proc. Ubicomp'17*, 1 (3): 1–24, 2017. ISSN 24749567. https://doi.org/10.1145/3130959.

M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool. Incremental learning of random forests for large-scale image classification. *IEEE Trans. Pattern Ana. Mach. Intell.*, 38 (3): 490–503, 2016. ISSN 0162-8828. https://doi.org/10.1109/TPAMI.2015.2459678.

R. L. Rivest, L. Adleman, M. L. Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4 (11): 169–180, 1978.

A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *ICCVW'09*, pages 1393–1400. IEEE, 2009. ISBN 978-1-4244-4442-7. https://doi.org/10.1109/ICCVW.2009.5457447.

V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*, pages 2576–2582, 2017.

T. Sztyler, and H. Stuckenschmidt. Online personalization of cross-subjects based activity recognition models on wearable devices. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom) 2017* Mar 13 (pp. 180–189). IEEE.

P. Vepakomma, D. De, S. K. Das, and S. Bhansali. A-Wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities. In *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, pages 1–6. IEEE, 2015.

J. Wang, Y. Chen, S. Hao, W. Feng, and Z. Shen. Balanced distribution adaptation for transfer learning. In *ICDM*, pages 1129–1134, 2017.

J. Wang, Y. Chen, L. Hu, X. Peng, and S. Y. Philip. Stratified transfer learning for cross-domain activity recognition. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2018a.

J. Wang, V. W. Zheng, Y. Chen, and M. Huang. Deep transfer learning for cross-domain activity recognition. In *ICCSE*, page 16, 2018b.

J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119: 3–11, 2019.

J. M. Wardlaw, E. E. Smith, G. J. Biessels, C. Cordonnier, F. Fazekas, R. Frayne, R. I. Lindley, J. T O'Brien, F. Barkhof, O. R. Benavente, et al. Neuroimaging standards for research into small vessel disease and its contribution to ageing and neurodegeneration. *The Lancet Neurology*, 12 (8): 822–838, 2013.

J. Wen, J. Indulska, and M. Zhong. Adaptive activity learning with dynamically available context. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–11. IEEE, 2016.

H. Xu, Z. Yang, Z. Zhou, L. Shangguan, K. Yi, and Y. Liu. Indoor localization via multi-modal sensing on smartphones. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 208–219. ACM, 2016.

J. B. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI, Buenos Aires, Argentina*, pages 25–31, 2015.

Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10 (2): 12, 2019.

S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher. DeepSense: A unified deep learning framework for time-series mobile sensing data processing. In *WWW*, pages 351–360, 2017.

M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *6th International Conference on Mobile Computing, Applications and Services*, pages 197–205. IEEE, 2014.

L. Zhang, X. Wu, and D. Luo. Recognizing human activities from raw accelerometer data using deep neural networks. In *IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 865–870, 2015.

M. Zhao, S. Yue, D. Katabi, T. S. Jaakkola, and M. T. Bianchi. Learning sleep stages from radio signals: A conditional adversarial architecture. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4100–4109. JMLR. org, 2017.

Z. Zhao, Y. Chen, J. Liu, Z. Shen, and M. Liu. Cross-people mobile-phone based activity recognition. In *IJCAI*, volume 11, pages 2545–2550, 2011.

Z. Zhao, Z. Chen, Y. Chen, S. Wang, and H. Wang. A class incremental extreme learning machine for activity recognition. *Cogn. Comput.*, 6 (3): 423–431, 2014. ISSN 1866-9956. https://doi.org/10.1007/s12559-014-9259-y.

Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.

Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang. Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services*, 14 (4): 352–375, 2018.

Social Network Analysis for Disinformation Detection

Aviad Elyashar, Maor Reuben, Asaf Shabtai, and Rami Puzis

1 Introduction

Since the seventeenth century, journalists have informed people about issues and events they need to know about. In many cases, this information influences the decisions people make every day [1]. However, in the last decade, journalism has struggled with a cultural change as news distribution has increasingly moved online [2]; this move is the result of technological changes and the increased use of mobile devices and instant messaging. Another reason for this shift is associated with the interactive capabilities of online news, such as the ability to easily click on an online story, share it, and post a comment about a given story [3].

Along with the growth of online news, many non-traditional news sources (e.g., blogs) have evolved in order to respond to users' "appetite for information." However, in many cases, these sources are operated by amateurs whose reporting is often subjective, misleading, or unreliable [4]. This everyone is a journalist phenomenon [5], together with the flood of unverified news and the absence of quality control procedures to prevent potential deception, has contributed to the increasing problem of fake news dissemination [6].

The spread of misinformation, propaganda, and fabricated news has potentially harmful effects and a significant impact on real-world events [7]. In recent years, it has weakened public trust in democratic governments and their activities, such as the "Brexit" referendum and the 2016 U.S. election [8]. World economies are also not immune to the impact of fake news; this was demonstrated when a false claim regarding an injury to President Obama caused the stock markets to plunge (dropping 130 billion dollars) [9]. In recent years, due to the threats to democracy,

A. Elyashar · M. Reuben · A. Shabtai · R. Puzis (✉)
Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel
e-mail: aviade@post.bgu.ac.il; maorreu@post.bgu.ac.il; shabtai@bgu.ac.il; puzis@bgu.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_30

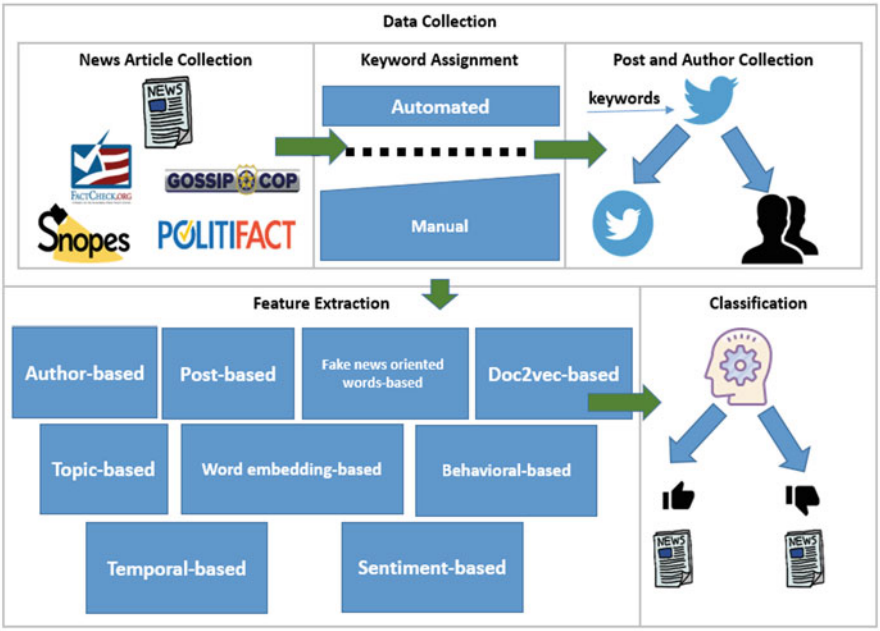


Fig. 1 The proposed method

journalistic integrity, and economies, researchers have been motivated to develop solutions for this serious problem [8]. Researchers have addressed this issue by proposing approaches for the detection of fake news based on natural language processing [10], investigating the diffusion of news [11], etc. A few papers have attempted to detect fake news solely using social context features [12].

In this paper, we present an automated system for news classification based on online social media (OSM) features. In Sect. 3, we describe our process for collecting the training set, gathering the supporting information from OSM, and constructing a classifier that differentiates between true and false claims using OSM features. This process is depicted in Fig. 1. In contrast to the related work presented in Sect. 2, the proposed approach is both automated and relies on the content-based retrieval of relevant social discussions.

Cross-validation using the *Fake News* dataset which consists of claims from fact-checking websites (Sect. 4) shows that individuals are highly aware of fake news and provide honest feedback on claims disseminated online in their posts (Sect. 5.2). Strengthened by other features, such feedback on claims can be used to effectively classify the claims as true or false, as shown by the classifier's performance: accuracy of 0.76 and an area under the receiver operating characteristic curve (AUC) of 0.89.

In order to facilitate the validity of the results, we have published the fake news dataset and a ready to use analytic process implemented in Python.[1]

2 Related Work

The proliferation of fake news worldwide has motivated researchers to explore solutions to address the problem of fake news dissemination [12]. Many of the previously proposed methods attempted to detect fake news by using natural language processing [8] or investigating the diffusion of news [11].

2.1 News Detection Based on Diffusion Analysis

In 2014, Friggeri et al. [13] examined rumor propagation on Facebook through rumor cascades. They collected 4761 claims from *snopes.com* and constructed rumor cascades based on the tree of Facebook reshares and the original post which contained a link to *Snopes* website. The authors found that while some rumors proceeded to flourish on Facebook, the true rumors were found to be the most viral and had the largest cascades.

In 2018, Vosoughi et al. [11] investigated the differences between the diffusion of verified true and fake claims using a dataset of rumors disseminated on Twitter from 2006 to 2017. They collected about 126,000 labeled claims from six fact-checking websites. For each claim, they collected original tweets that contained links to these claims published on the fact-checking websites. Then, they built rumor cascades based on the original tweets and their retweets and analyzed those cascades based on their depth, size, breadth, etc. The authors found that fake claims on Twitter diffused significantly farther, faster, deeper and more broadly than true claims. Unlike Friggeri et al. and Vosoughi et al., we collected tweets relevant to the given claim based on keywords. The collection of posts based on links solely may cause to lose a great number of posts that are associated with the given claim but do not include a link.

2.2 Fake News Detection Based on Sources

In 2017, Elyashar et al. [14] measured the authenticity of online discussions within OSM based on the similarity of users participating in the online discussion to known abusers and legitimate accounts.

[1] Will be published after acceptance.

In 2017, Volkova et al. [15] presented linguistically infused neural network models in order to classify tweets containing news as suspicious or verified. Their model combined tweet text, social graphs and linguistic cues for classification. For evaluation, the authors labeled specific sources as suspicious and verified.

Recently, Grinberg et al. [16] analyzed the exposure of fake news among a group of U.S. voters on Twitter during the 2016 election. The authors linked the two entities by pairing users and voters based on an exact match of names and locations from the text of Twitter profiles and voter registration records. Surprisingly, they found that the majority of political news across all political groups is still extracted from popular and authentic news sources. Therefore, fake news may not be more viral than real news.

The problem with these approaches is that they are too inclusive. Stating that all of the posts distributed by a specific source are fake or true is too binary and in most cases does not mesh with reality where a user might distribute fake news in one domain and the truth in another.

2.3 Detecting Fake News Based on Online Social Media Features

Online social media platforms offer unique features that can be used for news classification. In 2011, Castillo et al. [17] estimated the credibility of a given set of tweets related to 2500 news events using a deprecated service called Twitter Monitor. Then, they used evaluators from Mechanical Turk[2] in order to differentiate between news about a specific event and comments or conversation. Later, they proposed a machine learning classifier that estimated the credibility of each labeled topic based on the following type of features: message-based features (e.g., message length, whether the message contains a hashtag), user-based features (e.g., number of followers, followee), topic-based features (e.g., the fraction of tweets that contain URLs), and propagation-based features (e.g., depth of the retweet tree). Their best performing classifier achieved precision and recall in the range of 70–80%. As opposed to our fully automated approach, their method of differentiating events and unrelated comments is dependent on human annotators. We extracted the keywords automatically.

In 2015, Zhou et al. [18] demonstrated a real-time news certification system on Sina Weibo[3] using the keywords of an event in order to gather related microblogs. Then, they built an ensemble model that combined user-based, propagation-based, and content-based features and evaluated the proposed model on a small dataset of 146 claims. Their model obtained accuracy of approximately 0.8. As opposed to our work, Zhou et al. mainly focused on the proposed framework for real-time

[2] https://www.mturk.com/.

[3] https://www.weibo.com/.

news certification rather than on feature analysis. Furthermore, our analysis was demonstrated on approximately three times more claims.

In 2018, Wang et al. [19] presented an event adversarial neural network (EANN) system for fake news detection. EANN uses textual and visual features extracted from the posts' content for real-time fake news detection. Their approach focused on content-based features only, in contrast to our method which analyzes a large range of features.

Recently, Monti et al. [20] proposed studying fake news propagation patterns by exploiting geometric deep learning, a novel class of deep learning methods designed to work on graph-structured data. They demonstrated their approach on a dataset collected from Twitter and obtained an AUC of 0.93. Their limitation lies in their data collection. In contrast to our study in which tweets were collected based on keywords, Monti et al. collected tweets based on the sources distributed within fact-checking websites. Given this, they will be unable to predict whether a given story is false or true without these sources.

Also, Ma et al. [21] collected 778 claims from Snopes. Afterwards, they collected microblog posts from Twitter and Sina Weibo by extracting keywords from the suffix of the Snopes URL. In total, they collected more than million posts from Twitter and more than 3.8 million posts from Sina Weibo. They used RNN-based methods with three widely used recurrent units, tanh, LSTM and GRU for detecting rumors. Their methods performed significantly better than the state of the arts.

3 Methods

In this paper, we propose an automated approach for detecting fake news based on features extracted from posts and the authors participating in online discussions related to given news. In this section, we provide a comprehensive description of the proposed method, from data collection and feature extraction to news classification (see Fig. 1).

3.1 Data Collection

This stage includes three main sub-stages: claim collection, keyword assignment, and post and author collection according to the assigned keywords.

3.1.1 Claim Collection

First, in order to train an ML classifier, we collect claims for estimation. These claims can come from a variety of sources, including crawlers which collect claims from fact-checking websites, public news services, etc.

3.1.2 Keyword Assignment

After the collection of claims has been completed, the next step is to collect relevant posts and authors that are associated with them.

To the best of our knowledge, there are two ways to collect relevant posts for a given claim. One of the methods focuses on the sources that distributed the claims. Monti et al. [20] used the sources' headlines that exist in fact-checking websites in order to collect tweets. Vosoughi et al. [11] collected tweets that contained links to the given claims. We believe that both approaches have major drawbacks: Collecting tweets based on sources can be problematic in cases for which there are no sources. Moreover, the tweets with a link to the claims are a subset of all of the tweets that are relevant to the claims. Due to these drawbacks, we made the decision to tie a given claim to its corresponding tweets based on keyword assignment. We believe that this is the most effective method for collecting the maximal number of relevant posts for a given claim.

Keyword assignment can be an automated or manual process. Automated keyword assignment can utilize part of speech (POS) tagging to generate sets of keywords for each claim. Based on the heuristics suggested by [22], we narrow down the text to the following candidates: nouns, adjectives, adverbs, and numbers. The POS tagging words are prioritized as follows:

$$number \leq adverb \leq adjective \leq noun$$

Then, the K first words from the candidates are selected as input keywords. In the case of manual keyword assignment, human annotators are employed. An annotator is a person who assigns sets of keywords manually, based on his/ her expertise, in order to find the most appropriate sets of keywords for retrieving relevant posts.

Manual keyword assignment guidelines for relevant post retrieval using OSM search engines are provided below.

First, the annotator must read the given claim in order to understand the claim's subject. In many cases, reviewing the title and description is sufficient, but in other cases, the annotator must read the full article.

Second, some preprocessing is required, in which the annotator performs stemming and removes stop words from the claim's title and description.

Third, the annotator manually assigns sets of keywords that express the main idea of the claim. Similar to [23], we recommend including at least three to five different sets of keywords for each claim. For example, for the false claim "Pamela Anderson passed away in March 2018," the sets of keywords may be "Pamela Anderson March 2018" or "Pamela Anderson passed away."

Fourth, in many cases, in order to expand the context of the retrieved posts, annotators also use synonyms as keywords [24]. This facilitates the retrieval of a large number of posts relevant to a given claim. In the example mentioned above, we can change the claim slightly, referring to Pamela Anderson by her first name (Pamela) instead of her full name and using the term "died" instead of "passed

away." Thus, additional sets of keywords may be: "Pamela died," and "Pamela Anderson death."

Fifth, after assigning the sets of keywords, the keywords must be evaluated to determine whether they reflect the claim can be used to retrieve a sufficient number of relevant posts. In order to test this, the annotator queries the set of keywords manually using the OSM search engine. He/ she should review the retrieved posts and assess their relevance. In cases in which there are less than twenty posts, it might be wise to use additional synonyms as keywords (see Algorithm 1).

Algorithm 1 Manual keyword assignment

```
Read claim's title and description.
    If necessary, read the full claim
Remove stop words and perform stemming.
Assign keywords that express the main idea of the claim.
Provide 3–5 alternative sets of keywords.
    Use synonyms based on common knowledge.
Query the OSM search engine using the different sets of keywords.
    Read a few of the retrieved posts.
    Check relevance.
    Record the number of posts retrieved.
```

3.1.3 Post and Author Collection

After establishing the set of keywords assigned to each claim, we use those keywords in order to collect posts and authors from available public OSM services.

3.2 Feature Extraction

Similarly to Shu et al. who suggested that social context features may provide useful information that can help infer the veracity of claims [12], we extract the following features:

3.2.1 Author-Based Features

Group-level author features are an extension of individual-level features [12]. Individual-level features are extracted to infer the credibility of an author using various aspects of author demographics, such as registration age, number of followers, number of followees, number of distributed tweets published by the user, etc. [17]

As opposed to individual-level features, group-level author features encompass overall characteristics of groups or communities of authors related to the claims. [25] The main idea behind these features is that false and true news promoters may have unique characteristics as a community [12].

The group-level author features are calculated as a Cartesian product between the individual-level features (e.g., number of followers, number of followees, number of distributed posts, etc.) and aggregation functions (e.g., min, max, average, and standard deviation). For example, the "average number of followers" feature depicts the average number of followers who follow the authors who participated in the online discussion related to a given claim.

3.2.2 Post-Based Features

OSM platforms encourage users (authors) to express their opinions and emotions using posts distributed by these users [18]. Therefore, extracting these kinds of features can help identify potential deceptive claims according to the reactions expressed by the general public [12]. Post-based features are calculated as a Cartesian product between the individual post features (e.g., number of retweets and number of favorites) and aggregation functions (e.g., min, max, average, etc.) For example, the "max number of retweets" feature depicts the maximal number of retweets a post, related to a given claim can have.

3.2.3 Fake News Oriented Word-Based Features

In many cases, the revenue of OSM websites is not based on subscriber charges but instead is based on the advertisements that appear on the websites [26]. This results in a significant amount of competition among OSM outlets that vie for readers' attention and clicks, which increase the income of OSM websites. Therefore, in order to attract users and encourage them to visit a specific website and click on a given claim, website administrators may use a variety of techniques, including the use of eye-catching headlines along with links to the article [26].

In our approach, we use a fixed set of common clickbait phrases that include exaggeration, contradictions, etc. provided by Chakraborty et al. [26] for the given posts. For each post, we count the number of occurrences of each word contained in the fixed set of clickbait words. Afterwards, for each claim, we calculate the average number of occurrences of these words in the posts associated with the given claim.

Moreover, we create a fixed set of fake news oriented words; the words were collected by inspecting tweets associated with fake claims (words such as "fake," "liar," etc.)

3.2.4 Word Embedding-Based Features

Word representation is a well-known natural language processing paradigm, and a typical word representation approach represents words as vectors. These vectors are derived from several training methods extracted from neural network language modeling [27], and they are used to capture the semantic properties of words.

Our approach uses two pre-trained language models known as word embedding provided by GloVe [28]. In this research, we use one model, which was trained on about six billion words from the 2014 Wikipedia and English Gigaword Fifth Edition corpora [29] and consists of 300 dimensions. The second model consists of 200 dimensions and was trained on about 27 billion tweets from Twitter.

In this study, for each claim, we build an aggregated GloVe word embedding vector based on the words included in the posts associated with the given claim. This means that each claim was represented by the bag-of-words model of all associated tweets related to the given claim. Then, for each word in a tweet, we search for the appropriate vector in the GloVe word embedding model. Lastly, we aggregate all those vectors to a single aggregated vector by exploiting pooling techniques. Often pooling techniques choose the highest average lowest value in each dimension from a set of vectors. In our approach, we use three aggregation functions (minimal, maximal, and average) in order to create three representations for each claim. The maximal representation is the most common since it should capture the most prominent words that exist in the tweets describing the claim [30]. The minimal and average representations have also been used for sentiment classification [31]. In the last stage, we extract each dimension that exists in the aggregated vector representation as a feature.

This means that for each claim we build three aggregated word embedding models. In the last stage, we extract each dimension that exists in the aggregated vector representation as a feature.

3.2.5 Doc2vec-Based Features

In 2014, Le and Mikolov [32] extended the idea of feature representation of words as a basic unit to variable-length pieces of texts, such as sentences, paragraphs, and documents.

In our approach, we treat each associated post as a basic unit, and the doc2vec model is trained on the associated posts. Afterwards, we aggregate all of these vectors to a single aggregated vector which represents the given claim. The aggregation functions are: average, minimum, and maximum. As in the previous case, we extract each dimension that exists in the final aggregated vector representation as a feature.

3.2.6 Topic-Based Features

Topic-based features are a unique post feature type that reflects the social response from the general public. Topic features can be extracted using topic models, such as latent Dirichlet allocation (LDA) [33].

In our approach, for each claim, we collect all of the relevant corresponding posts and divide them into 10 topics. Each post has a probability of being related to the given topic. We assign each post to the topic which has the maximal probability. Later, we collect all of the maximal probabilities for all of the posts associated with a given claim and run aggregation functions, such as minimum, maximum, average, and standard deviation.

3.2.7 Sentiment-Based Features

In many cases, sentiment analysis can assist in the detection of deceptive news by detecting unintended emotions expressed by deceivers [34].

3.2.8 Temporal Features

In many cases, the spread of fake news within OSM platforms involves unique temporal patterns that differ from those of true news [12].

According to our approach, for each claim, we calculate the time intervals in which the authors created or distributed associated posts. We aggregate those intervals by the functions: average, minimum, and maximum, skewness, and kurtosis.

3.2.9 Behavioral Features

One of the approaches for detecting fake news suggests detection based on the suspicious behavior of the authors that distributed misinformation [12].

In our approach, for each claim, we calculate the following features per author: the average minutes between posts, the average posts per day, and the number of retweets.

3.2.10 Graph-Based Features

This approach focuses on network patterns. In most cases, graph-based features are extracted by constructing networks among the authors that distributed posts related to a given claim [12].

In our approach, for each claim, we create graphs based on follower and followee connections. Later, we calculate network features as a Cartesian product between

network measures (e.g., degree, betweenness, and closeness centrality, page rank, etc.) and aggregation functions (e.g., maximum, median, minimum, and standard deviation).

3.3 Classification

After the feature extraction phase has been completed, we construct a machine learning classifier using the extracted features, in order to differentiate between false and true claims.

4 Data

In this section, we describe the data collected according to the process described in Sect. 3.1.

4.1 Claims

First, we collected a set of 386 labeled claims from several fact-checking websites (see Table 1).

The claims were published on the fact-checking websites between June 1997 and December 2018.

Human fact-checkers for the websites investigated each claim and provided verdicts, such as "fact," "mostly true," "mostly false," etc. Here, we consider "true," "mostly true," and "fact" as labels for true claims, and "barely true," "mostly false," "false," "pants on fire," and "scam" as labels for false claims. Each downloaded claim includes descriptive attributes, such as title, description, verdict, verdict date, and a link to the analysis report supporting the verdict.

Table 1 Fact-checking websites

	Total	True	False	Tweets	Authors
snopes.com	311	165	146	554,886	432,140
politifact.com	36	8	28	118,366	91,883
factcheck.org	17	0	17	29,148	20,705
gossipcop.com	10	0	10	11,532	6,535
thejournal.ie	1	0	1	1224	1075
usnews.com	9	9	0	118,327	81,660
hoax-slayer.net	2	0	2	1184	916
Total	386	182	204	834,667	634,914

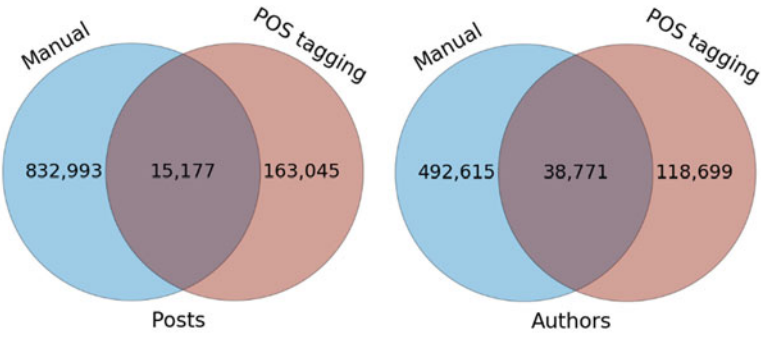


Fig. 2 Author and tweet distributions using manual keyword assignment and automated POS tagging keywords

4.2 Keywords

In this study, we collected two datasets: one dataset collected using manual keyword assignment and the second dataset that was collected by an automated part of speech (POS) tagging keywords. The dataset extracted based on the keywords assigned manually contains 832,993 tweets and 492,615 authors. The POS tagging keyword dataset contains 163,045 tweets and 118,699 authors. The two datasets have an overlap of 15,177 tweets and 38,771 authors (see Fig. 2).

4.3 Posts and Authors

With the keywords assigned for each claim, we collected tweets using the public Twitter API. We chose Twitter because with more than 321 million active users at the end of 2018 [35], it is one of the largest and most popular OSM networks worldwide.

In total, we retrieved 834,667 tweets published by 634,914 authors. The number of retrieved tweets ranged from 20 to more than 40,000 per claim (see Fig. 3). The number of authors ranges from 14 to more than 32,000 per claim.

5 Evaluation

5.1 Experimental Setup

As part of our experiments, we constructed news classifiers that differentiate between false and true claims based on the proposed feature subsets. In total, we extracted 1826 features representing these claims (see Table 3).

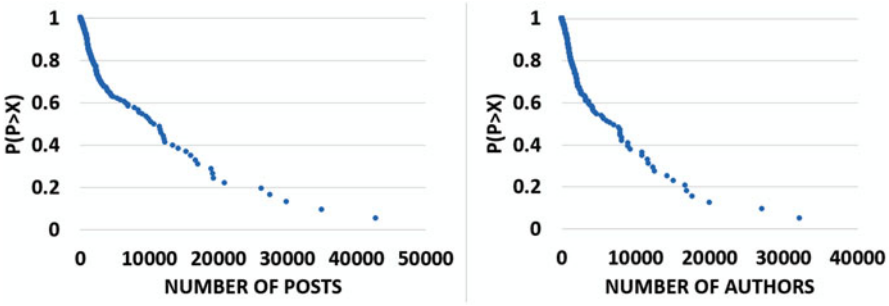


Fig. 3 Distribution of the number of posts (left) and the number of authors (right) across claims

After the feature extraction phase, we trained several machine learning classifiers on the *Fake News* dataset presented in Sect. 4. The ML algorithms used were XGBoost, Random Forest, AdaBoost, and Decision Tree. The *Fake News* dataset was sorted according to the claims' verdict date. The data was divided into train and test sets which contained 300 and 86 claims, respectively. Each classifier was trained with multiple sets of features sorted by the feature importance score (5, 10, and 15 top features). The performance of the classifiers was evaluated in terms of the area under the receiver operating characteristic curve (AUC), accuracy, F1, precision, and recall.

5.2 Results and Discussion

In this section, we present the results of our study based on the utilization of the feature subsets for news classification (see Sect. 3). The results are presented as follows: First, we analyze the features' effectiveness based on the information gain score. Then, based on the number of features (5, 10, 15), we find the best classifier and discuss its performance.

5.2.1 Feature Importance

To obtain an indication of the usefulness of the various features, we analyzed their importance using Weka's information gain attribute selection algorithm. These attributes were then fed into Weka [36], a popular machine learning software suite written in Java and developed at the University of Waikato, New Zealand.

According to the information gain score, we can see that while some attributes are very indicative, many other features are less useful for news classification. Among the 1824 extracted features, only 163 features were found to be indicative (obtained an information gain score higher than zero). Among these features, 133 features

were word embedding-based, 22 were user-based, six were fake news oriented words-based, and there was one of the doc2vec- and behavioral-based features.

In this research, we were interested in identifying the optimal word embedding model to use while working with data crawled from the OSM. Our intuition led us to believe that it is not possible to identify a single model for all cases and that in fact, it likely depends on the data used. Therefore, in this study in which we are working with tweets, we use a word embedding model trained on Twitter.

In order to address this, we compared the indicative features extracted based on word embedding models trained on Twitter and Wikipedia. Surprisingly, there is no advantage for features extracted based on Twitter although the corresponding posts were collected from it. In addition, the differences between the information gain scores of the indicative features extracted based on the models trained on Twitter and those trained on Wikipedia were not to be significant ($p = 0.967$).

With regard to the author-based features, most of the features were found to be indicative according to the information gain score. Moreover, among the top 10 indicative features, six features were author-based. These features are related to the attributes of authors that distributed tweets related to a given claim, such as the number of followers, lists, followees, etc. Regarding fake news oriented words-based features, we can see that the indicative features are related to words that contradict the given claim, such as the words "fake" and "liar."

The 10 attributes with the highest rank are presented in Table 2.

As can be seen, the most significant feature among all of the extracted features is the fraction of the word "fake" in posts related to the claim which obtained an information gain score of 0.12. The number of occurrences of this word was found to be useful for detecting fake news (information gain score of 0.08). Moreover, we can see that the fraction of the word "fake" is an average of six times higher in false claims (0.036) than it is in true claims (0.006); the occurrence of this word is three times higher in fake claims (31.28 versus 11.1). These results demonstrate the power of "the crowds" for helping detect false and true claims given the awareness of skeptical authors who participate in online discussions and raise doubt and call attention to deceptive content included in a claim by using words that contradict the claim.

With regard to user-based features, there are a few features, such as the number of lists, followers, and followees, that were found to be essential for detecting fake news. We can see that authors that distributed true claims registered to more lists (250.1) than authors that distributed false news (151.45) and have two times more followers (33,275.86 versus 16,573.61 followers) and more followees (2825.5 versus 2421.05). It is important to note that the differences between the top nine features were found to be statistically significant.

5.2.2 Best Classifier

After trying many combinations of machine learning algorithms and feature subsets, we determined that the best performing classifier on the training set was the

Table 2 Top 10 features according to information gain

Rank	Feature	Group	Info gain	True claim		False claim	
				AVG	StDev	AVG	StDev
1	Fraction of the word "fake"	Oriented	0.12	0.006	0.016	0.036	0.139
2	StDev num of lists	User	0.11	1984.75	2125.34	910.47	1487.63
3	StDev num of followers	User	0.1	395,793.5	564,206.1	162,412.4	386,217.7
4	Min GloVe Twitter 78d	Embedding	0.093	−1.266	0.17	−1.308	0.238
5	Avg num of lists	User	0.09	250.1	205.94	151.45	163.45
6	Avg num of followers	User	0.09	33,275.86	44,988.17	16,573.61	28,245.4
7	Num of the word "fake"	Oriented	0.08	11.1	44.47	31.28	86.05
8	StDev num of followees	User	0.08	10,738.12	7351.74	7455.59	6625.41
9	Average number of followees	User	0.074	2825.5	1216.28	2421.05	1369.3
10*	Min GloVe Wikipedia 97d	Embedding	0.073	−1.11	0.22	−1.148	0.28

* Difference between values of the features in the positive examples and in the negative examples is statistically significant ($p < 0.05$) except feature 10

XGBoost with 10 features, 30 estimators a maximal depth of six, and a learning rate of one. This classifier resulted in an AUC and the accuracy of one on the training set. Evaluating its performance on the test set, we obtained an AUC of 0.89, accuracy of 0.76, F1 of 0.6, precision of 0.5, and recall of 0.76. The classifier that performed the best on the test set was the XGBoost with 15 features, 30 estimators, a maximal depth of three, and a learning rate of 0.2; in this case, this classifier obtained an AUC of 0.89, accuracy of 0.76, F1 of 0.63, precision of 0.5, and recall of 0.86.

5.3 Feature Subset Comparison

In order to evaluate the effectiveness of the proposed feature subsets, we performed a comparison of various combinations of feature subsets and machine learning algorithms. For each feature subset, we trained four machine learning algorithms (XGBoost, Random Forest, AdaBoost, and Decision Tree) on the 5, 10, and 15 most influential features in the subset. The results of the best classifiers for each feature subset are summarized in Table 3.

The results show that the best performance was achieved by fake news oriented words feature subset. This finding emphasizes the power of authors who doubt suspected claims. Moreover, the second best classifier was trained on word embedding features. This finding implies that authors who discuss false and true claims speak differently and these differences can be detected using word embedding. Surprising, the classifier which was trained solely on Wikipedia outperformed slightly better than the classifier that was trained on the Twitter corpus.

The third best classifier was a mixed model that was trained on behavior, syntax, sentiment, and topic-based features. Based on this result, we can conclude that features based on behavior (e.g., the average number of posts per day and average number of retweets), syntax (e.g., the average number of links, average post length, average user mentions), sentiment (e.g., average negative, and positive emotions), and topic-based features are useful for news classification.

5.3.1 Number of Posts per Claim

In this study, one of the questions that most interested us was whether the number of relevant posts retrieved for a given claim affects the classifier's ability to detect fake news. In order to answer this, we trained four algorithms: Random Forest, XGBoost, AdaBoost, and Decision Tree with 10 and 15 features. The classifiers were trained using two different methods; the first used 10-fold cross-validation, while the second was based on the descending order of the number of posts related to each claim. The training set was divided equally by the targeted label, e.g., 10–10 means 10 records of labeled true and false claims. The training size ranges from 20 records (10–10) to 360 records (180–180). The 10-fold cross-validation experiments were run 10 times, whereas the experiments that include the descending order of the

Table 3 The performance of the best classifiers per feature subset

Subset	Total features	Algorithm	Features	AUC	Acc.	F1	Pr.	Rec.
Fake news words	**24**	**XGBoost**	**10**	**0.76**	**0.71**	**0.73**	**0.72**	**0.75**
Author- and post-based	50	Random forest	15	0.72	0.66	0.68	0.67	0.69
GloVe word embed	1500	Random forest	15	0.73	0.67	0.69	0.68	0.7
GloVe Twitter word embed	600	XGBoost	15	0.74	0.66	0.68	0.68	0.69
GloVe Wiki word embed	900	XGBoost	15	0.74	0.69	0.70	0.71	0.69
Doc2vec-based	100	Random forest	10	0.66	0.63	0.67	0.64	0.69
Behavior + Topic + Syntax + Sent'	19	Random forest	15	0.72	0.69	0.72	0.7	0.73
Graph-based	59	AdaBoost	5	0.66	0.67	0.71	0.66	0.76
Temporal-based	72	Random forest	5	0.63	0.6	0.64	0.62	0.66
All features	1824	XGBoost	15	0.72	0.67	0.71	0.67	0.76

number of posts were run only once. We found that classifiers which were trained using the method based on the descending order of the number of posts related to each claim outperformed the random selection of records. Given this, we conclude that the number of associated posts related to a given claim affects the classifier's performance.

6 Ethical Considerations

Collecting information from online social media has raised ethical concerns in recent years. To minimize the potential risks that may arise from such activities, this study followed the recommendations presented by [37], which deal with ethical challenges regarding OSM and Internet communities.

In this study, we used the public service of Twitter REST API for two purposes: first, to collect tweets associated with given news topics and, second, to identify the authors who distributed those tweets. The Twitter REST API collects the information of accounts that agree to share their information publicly. Moreover, the research protocol was approved by the Ethics Committee of the Anonymous University.

7 Conclusion and Future Work

In this paper, we proposed an automated approach for detecting fake news based on OSM posts and authors participating in related online discussions. For this purpose, we collected online discussions (posts) and identified the authors associated with the given claims using automated keyword assignment. Based on these posts and authors, we extracted features with respect to these claims. Next, we trained a machine learning classifier which differentiates between false and true claims. We demonstrated the proposed approach on a novel fake news dataset collected from Twitter. Based on our results and analysis, we can make the following observations: First, we showed that the posts and authors on OSM platforms can be utilized in order to detect fake news. The best classifier which its features are post and author-based obtained an AUC and accuracy of 0.89 and 0.76, respectively, on the test set (see Sect. 5.2.2).

Second, analyzing the importance of the proposed features (e.g., the fraction of the word "fake" within the posts) emphasizes the power of authors participating in online discussions in identifying fake news.

In addition, objective author-based features, such as standard deviation and average of number of lists and followers were found to be essential for the detection of fake news. This result reinforces the conclusions of Zhou et al. [18] who detected rumors in Sina Weibo based on personal user features. Furthermore, based on the results of word embedding-based features, we conclude that (1) there is no

advantage to using a GloVe word embedding model that has been trained on Twitter when analyzing tweets (see Sect. 5.2), and (2) authors speak differently when discussing fake or true news. This finding reinforces the conclusion of Wang [38] who detected fake news using word vectors constructed based on the claim's content. Lastly, we found a positive correlation between the news classifier's performance and the number of posts associated with the given claim. In future work, we plan to demonstrate the proposed process on different OSM platforms, such as Reddit, Quora, etc.

References

1. Benjamin D Horne and Sibel Adali. This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Eleventh International AAAI Conference on Web and Social Media*, 2017.
2. John Pavlik. The impact of technology on journalism. *Journalism studies*, 1(2):229–237, 2000.
3. Pablo J Boczkowski and Eugenia Mitchelstein. How users take advantage of different forms of interactivity on online news sites: Clicking, e-mailing, and commenting. *Human Communication Research*, 38(1):1–22, 2012.
4. Leonard Downie and Michael Schudson. The reconstruction of American journalism. *Columbia Journalism Review*, 19, 2009.
5. Yuan Zeng. Danger, trauma, and verification: eyewitnesses and the journalists who view their material. *Media Asia*, 45(1–2), 2018.
6. Niall J Conroy, Victoria L Rubin, and Yimin Chen. Automatic deception detection: Methods for finding fake news. In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, page 82. American Society for Information Science, 2015.
7. Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36, 2017.
8. Xinyi Zhou, Reza Zafarani, Kai Shu, and Huan Liu. Fake news: Fundamental theories, detection strategies and challenges. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 836–837. ACM, 2019.
9. K Rapoza. Can 'fake news' impact the stock market? *Pridobljeno iz www. forbes. com/sites/kenrapoza/2017/02/26/can-fake-news-impact-the-stock-market/(9. 7. 2018)*, 2017.
10. Zhixuan Zhou, Huankang Guan, Meghana Moorthy Bhat, and Justin Hsu. Fake news detection via NLP is vulnerable to adversarial attacks. *arXiv preprint arXiv:1901.09657*, 2019.
11. Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.
12. Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36, 2017.
13. Adrien Friggeri, Lada Adamic, Dean Eckles, and Justin Cheng. Rumor cascades. In *ICWSM*, 2014.
14. Aviad Elyashar, Jorge Bendahan, Rami Puzis, and Maria-Amparo Sanmateu. Measurement of online discussion authenticity within online social media. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 627–629. ACM, 2017.
15. Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 647–653, 2017.

16. Nir Grinberg, Kenneth Joseph, Lisa Friedland, Briony Swire-Thompson, and David Lazer. Fake news on twitter during the 2016 us presidential election. *Science*, 363(6425):374–378, 2019.
17. Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM, 2011.
18. Xing Zhou, Juan Cao, Zhiwei Jin, Fei Xie, Yu Su, Dafeng Chu, Xuehui Cao, and Junqiang Zhang. Real-time news certification system on Sina Weibo. In *Proceedings of the 24th International Conference on World Wide Web*, pages 983–988. ACM, 2015.
19. Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. EANN: Event adversarial neural networks for multi-modal fake news detection. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 849–857. ACM, 2018.
20. Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*, 2019.
21. Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824, 2016.
22. Pengqi Liu, Javad Azimi, and Ruofei Zhang. Automatic keywords generation for contextual advertising. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 345–346. ACM, 2014.
23. Chengzhi Zhang. Automatic keyword extraction from documents using conditional random fields. *Journal of Computational Information Systems*, 4(3):1169–1180, 2008.
24. Ellen M Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69. Springer-Verlag New York, Inc., 1994.
25. Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. Automatic detection of rumor on Sina Weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM, 2012.
26. Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. Stop clickbait: Detecting and preventing clickbaits in online news media. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 9–16. IEEE, 2016.
27. Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL (2)*, pages 302–308, 2014.
28. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.
29. English Gigaword fifth edition. https://catalog.ldc.upenn.edu/LDC2011T07. Accessed: 2019-03-19.
30. Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *arXiv preprint arXiv:1805.09843*, 2018.
31. Duy-Tin Vo and Yue Zhang. Target-dependent twitter sentiment classification with rich automatic features. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
32. Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.
33. David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
34. Jeffrey T Hancock, Michael T Woodworth, and Stephen Porter. Hungry like the wolf: A word-pattern analysis of the language of psychopaths. *Legal and criminological psychology*, 18(1):102–114, 2013.

35. Twitter monthly active users. https://www.statista.com/statistics/274564/monthly-active-twitter-users-in-the-united-states/. Accessed: 2019-02-16.
36. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.
37. Yuval Elovici, Michael Fire, Amir Herzberg, and Haya Shulman. Ethical considerations when employing fake identities in online social networks for research. *Science and engineering ethics*, 20(4):1027–1043, 2014.
38. William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

Online Propaganda Detection

Mark Last

1 Introduction

Propaganda is characterized by deliberate selectivity and manipulation of information presented to the target audience [1]. Usually, the propaganda content is aimed at influencing peoples' opinions while trying to avoid the detection of its real agenda. Though the term *propaganda* has become widespread only in the twentieth century, the concept itself goes back to the ancient times. One of the earliest examples of political propaganda can be found in the Behistun inscription (sixth century BCE), where the Persian King Darius I tried to establish the legitimacy of his claim to Cyrus the Great's throne, to which he had no blood connection [2]. Five hundred years later, in the Ancient Rome, detailed guidelines for conducting a pre-election propaganda campaign were sent by Quintus Cicero to his brother Marcus Cicero who was running for the Roman Consulship. Here are some of his practical suggestions written in 64 BCE:

> Lastly, take care that your whole candidature is full of éclat, brilliant, splendid, suited to the popular taste, presenting a spectacle of the utmost dignity and magnificence. See also, if possible, that some new scandal is started against your competitors for crime or looseness of life or corruption, such as is in harmony with their characters [3].

All content delivery technologies that have replaced stone and papyrus in the course of centuries were promptly adopted as propaganda tools. Thus, printed books and films were extensively used by the Nazi propaganda machine for justification of their massive crimes against humanity. Music propaganda has also served the dictatorships around the world. For example, one of the most popular Soviet songs during the terrible years of Stalin's Great Terror (1936–1939) has been "Wide is my

M. Last (✉)
Ben-Gurion University of the Negev, Beer-Sheva, Israel
e-mail: mlast@bgu.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_31

Motherland," which claimed that there is no other country, where "a man can breathe so freely." Nowadays, the Internet has become a major medium for propaganda dissemination. Public websites, online forums, social network platforms, and multimedia channels are being actively misused by political leaders, authoritarian regimes, terrorist organizations, hate groups, and lone-wolf terrorists. In fact, online propaganda dissemination has a unique set of advantages over traditional mass media technologies (mainly, newspapers, radio, and TV). These advantages include:

- Negligible Publication Cost. Social media platforms enable and even encourage publication of almost any textual and multimedia content free of charge.
- Source Anonymity. The propagandist can easily conceal his/her real identity, location, political affiliation, etc. by exposing a fake profile or even pretending to be someone else. ID verification of every user is not required by most social media systems yet.
- Global Accessibility. On the Internet, the content can be delivered from anywhere to anywhere almost instantly. In most cases, propagandists from one country can easily target users in other countries.
- Content Personalization. Different propaganda content can be delivered to different users based on their specific characteristics.
- Low Signal-to-Noise ratio. As explained in the subsequent sections of this chapter, online propaganda is extremely hard to detect as it constitutes a very low percentage of the Internet traffic and often uses nearly the same vocabulary and symbols as legitimate web content.

This survey chapter is organized as follows. Section 2 reviews machine learning methods for detecting radical propaganda content disseminated by terrorist organizations and their supporters. In Sect. 3, we proceed with automated detection of propaganda campaigns on social media platforms. Open issues and directions for future research are summarized in Sect. 4.

2 Terrorist Content Detection

2.1 Online Terrorist Propaganda

Shortly after the invention of the World Wide Web (WWW) in the early nineties of the twentieth century, terrorist organizations have started using the Internet as an accessible and cost-effective propaganda infrastructure. For more than 20 years now, secure and non-secure web sites, online forums, and file-sharing services have been routinely used by terrorist groups for spreading their word, recruiting new supporters, communicating with their affiliates, and sharing knowledge on forgery, explosive preparation, and other "core" terrorist activities. The evidence of terrorists using the internet goes back as far as 1997. According to Jane's Foreign Report, "a full range of instructions for terrorist attacks, including maps, photographs, directions, codes and even technical details of how to use the bombs are being

transferred through the Internet" [4]. A few years after this report was published, the 9–11 hijackers have used the internet to communicate with each other, while staying completely unnoticed by the intelligence agencies [5]. As the German authorities have discovered after the tragic events of September 2001, the members of the infamous Hamburg Cell have actively used a computer for Internet research on flight schools and other topics of their interest [6].

According to MSNBC News Services [9], a Pentagon research team was monitoring more than 5000 Jihadist Web sites back in May 2006. Examples of such continuously active websites include https://saraya.ps/, https://www.palinfo.com/, and http://www.moqawama.org/, which are associated with Palestinian Islamic Jihad, Hamas, and Hezbollah, respectively. Unlike legitimate media portals, the URLs of terrorist websites are extremely volatile. For example, in 2006, the same three organizations were running their websites from the following addresses: http://www.qudsway.com, http://www.palestine-info.com/, and http://www.moqavemat.ir/. By October 2019, two of these three 2006 domains were offered for sale and one was just unavailable. Since terrorist propaganda websites, in general, try to avoid detection by frequently changing their web domains and URLs as well as the geographical locations of their web servers, there is a need to develop *automated methods* for detecting terrorist content in the massive amounts of regular web traffic [13].

Terrorist propaganda websites have proved themselves as an effective recruitment tool. There is a growing evidence that online terrorist content may cause Internet users, having no direct links to any terrorist group, to undergo a process of radicalization and then perform acts of terror on their own. Thus, a Spanish court has concluded that the deadly 2004 Madrid train bombings were carried out by a local Islamist cell inspired by an Islamic essay published on the Internet [10]. Similarly, a British Government report concluded that the July 7, 2005 bombings in London were a low-budget operation carried out by four men who had no direct connection to Al Qaeda and who obtained all the information they needed from the Al Qaeda sites on the Internet [11]. A recent study of 223 convicted United Kingdom-based terrorists [12] has found that in 61% of cases, online activity was related to the offenders radicalization and/or attack planning. According to the same study, lone-actor terrorists and extreme-right-wing offenders were more likely to engage in online learning.

2.2 Using Word Graphs for Terrorist Content Detection

Terrorist-generated content detection can be seen as a binary categorization problem, where the goal is to label a document or a complete website as either *terrorist* or *non-terrorist*. This categorization task has the following specific requirements [14]:

- High accuracy. A terrorist content detection system should be able to identify as many terrorist websites as possible while minimizing the amount of false alarms.

- Explainability. An automatically induced model should be subject to scrutiny by a human expert who may be able to enhance/correct the classification rules based on her/his own knowledge in the terrorist domain.
- Inference speed. Due to a huge amount of information that is being continuously posted and updated online, the model should be capable to process massive streams of web documents in minimal time.
- Multilinguality. The model induction methods should maintain a high performance level over web content in multiple languages used for terrorist propaganda. For example, Hamas official website ("The Palestinian Information Center") currently provides content in eight different languages (English, French, Arabic, Russian, Persian, Urdu, Turkish, and Malay).

Markov et al. [14] present a classification-based approach to multi-lingual detection and categorization of terrorist documents, which builds upon the graph-based text representation model developed by [15]. Unlike the traditional bag-of-words (BOW) model, this representation is based on the order of terms in a text document. Each unique term (keyword) appearing in the document becomes a node in the graph representing that document. Distinct terms (stems, roots, lemmas, etc.) can be identified by character n-grams normalization techniques. Each node is labeled with the term it represents. The node labels in a document graph are unique, since a single node is created for each distinct term even if a term appears in the document in several different contexts. Second, if a word a immediately precedes a word b, there is a directed edge from the node labeled as a to the node labeled as b. The ordering (contextual) information is particularly important for representing texts in languages like English and Arabic, where phrase meaning strongly depends on the word order. An edge is not created between two graph nodes if the corresponding words are separated by certain punctuation marks (such as periods). Word-based graph representations are language-independent: they can be applied to a tokenized text in any language without the need of applying any language-specific sentence parsing tools.

In the hybrid representation approach [14], terms (discriminative features) are defined as sub-graphs selected to represent a document already converted into a graph form. After the training documents are labeled by categories (e.g., *terrorist* vs. *non-terrorist*, a frequent sub-graph extraction algorithm is used for the identification of the most discriminative sub-graphs. Then all document graphs are represented as vectors of Boolean features (1—a selected sub-graph appears in the graph of a particular document; 0—otherwise). The resulting binary vectors can be used for inducing text categorization models by any standard classification algorithm.

2.3 Case Studies

In the first case study, Last et al. [14] built a corpus of 648 Arabic documents including 200 web pages downloaded from terrorist web sites and 448 web pages

from legitimate news websites. The corpus contained 47,826 distinct Arabic words (after normalization and stopword removal). Legitimate documents were taken from four popular Arabic news sites: Al Jazeera www.aljazeera.net/News, CNN in Arabic http://arabic.cnn.com, BBC News in Arabic http://news.bbc.co.uk/hi/arabic/news, and UN News in Arabic http://www.un.org/arabic/news. Each document was manually verified by a fluent Arabic speaker as not containing any terrorist propaganda content. Terror documents were downloaded from http://www.qudsway.com and http://www.palestine-info.com/ (calling itself The Palestinian Information Center, or PIC), which were associated, at that time, with Palestinian Islamic Jihad and Hamas, respectively, according to the SITE Institute web site (http://www.siteinstitute.org/). A human expert, fluent in Literary Arabic, has manually chosen 100 pages from each web site and labeled them as containing terrorist propaganda based on the entire content of each document rather than just occurrence of any specific keywords. A typical example of such document, including its translation into English, is shown in Fig. 1.

Using the popular C4.5 decision-tree algorithm with 100-nodes document graphs (representing 100 most frequent non-functional words in a document), which were converted into binary vectors, brought the highest classification accuracy of 98.5%, based on 10-fold cross-validation. The resulting decision tree is shown in Fig. 2. The tree contains five binary features: four features representing single-node sub-graphs (the words "The Zionist" in two grammatical forms, "The martyr," and "The enemy") and one two-node sub-graph ("Call [of] Al-Quds" in the document text, standing for the alias name of the Hamas web site). This simple decision tree can be easily interpreted as follows: if at least one of these five terms appears in an Arabic web document with a predefined minimal in-document frequency (at least two occurrences), it can be safely labeled as "terrorist (propaganda)." On the other hand, a document that contains none of these terms more than once should be labeled as "non-terrorist." Moreover, the induced model provides an important insight into Hamas propaganda, which tries to delegitimize the State of Israel by representing it

Quality Palestinian operations, the Zionist losses continued

Gaza correspondent of "Call of Al-Quds"

As a new Gregorian year has passed, the Zionist war has intensified the fiery of the Palestinian people on the one hand and of the forces of the resistance on the other hand. In response to the catastrophic failures, terrorist measures have followed and will follow in order to stop the intifada and defeat the Palestinian people and their continuous and renewed resistance.

العمليات الفلسطينيه النوعيه الخسائر الصهيونيه المستمره

غزه مراسل نداء القدس

سنه ميلاديه جديده مرت الحرب الصهيونيه تزداد شراسه الشعب الفلسطيني جهه ، وضد قواه المقاومه جهه اخرى. ردا الفشل الذريع منيت الاجراءات الارهابيه اتبعت وتتبع اجل وقف الانتفاضه الحاق الهزيمه الشعب الفلسطيني ومقاومته المستمره المتجدده.

Fig. 1 Example 1: Terrorist propaganda in Arabic

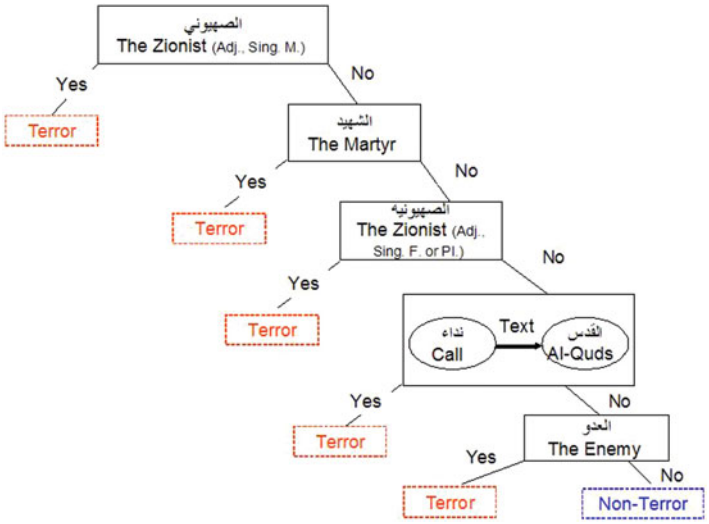


Fig. 2 C4.5 decision tree for classification of web pages in Arabic [14]

as a "Zionist entity" or just as an "enemy," its population as "Zionist settlers," etc. On the other hand, the website frequently praises the suicide bombers as "martyrs" (Shahids in Arabic) despite the fact that suicide is prohibited in Islamic law.[1]

The goal of the second case study was identifying the *source* of terrorist propaganda content. The experimental collection consisted of 1004 English documents obtained from the following two sources:

- 913 documents downloaded in 2006 from a Hezbollah web site.[2] These documents contained 19,864 distinct English words (after stemming and stopword removal). A typical example of such document is shown in Fig. 3.
- 91 documents downloaded in 2006 from a Hamas web site.[3] These documents contained 10,431 distinct English words.

Both these organizations are located in the Middle East with Hezbollah based in Lebanon and Hamas operating from Gaza Palestinian Territory. Making a distinction between the content provided by Hezbollah and Hamas is a non-trivial task, since these two Jihadi organizations are known to have close ties with each other resulting from their common cause and ideology.

The resulting decision tree is shown in Fig. 4. The tree contains four binary attributes: two attributes representing single-node sub-graphs (the words "Arab" and "PA"—Palestinian Authority) and two two-node sub-graphs (a hyperlink to

[1] And do not kill yourselves [or one another]. Indeed, Allah is to you ever Merciful [Qur'an 4: 29]. Source: http://www.dar-alifta.org/Foreign/ViewFatwa.aspx?ID=7149.

[2] http://www.moqawama.org/english/.

[3] www.palestine-info.co.uk/am/publish/.

Hizbullah draws massive crowd to mark "Israeli" withdrawal, Nasrallah: Resistance is a point of strength for Lebanon

BEIRUT: A quarter of a million Hizbullah supporters packed a square in the southern port city of Tyre Friday to mark the anniversary of "Israel's" withdrawal from South Lebanon in 2000. A day earlier, the Islamic resistance group had launched a 10-day campaign to collect funds for Palestine, which is facing a crippling Western aid boycott after the election of Hamas.

In a speech delivered on the occasion, Hizbullah chief Sayyed Hassan Nasrallah said the Lebanese must join forces to resolve the numerous issues facing the country.

"I advise the Lebanese to stop referring to the UN Security Council and [the international community]," he said. "Let us unite our efforts to resolve our problems."

...

The massive celebration was held near the former "Israeli" military headquarters in Tyre amid pressure to disarm from the UN Security Council.

However, Nasrallah insisted the resistance would continue fighting "Israel" until the Jewish (Zionist) state withdrew from the (occupied) Shibaa Farms and released all Lebanese detainees held in its prisons.

The cleric added that while "Israel" had defeated Arab armies for decades, its withdrawal from Lebanon in May 2000 "after incessant attacks" by Hizbullah marked a significant turning point.

"We have entered the phase of victory," he said. "After May 25, there would be no more `Nakba` and no more `Naksa,`" Nasrallah said in reference to the 1948 "Catastrophe" which led to the creation of "Israel" and the devastating 1967 Arab defeat. "[From this point forward] there will only be resistance, liberation and victory."

Fig. 3 Example 2: terrorist propaganda in English

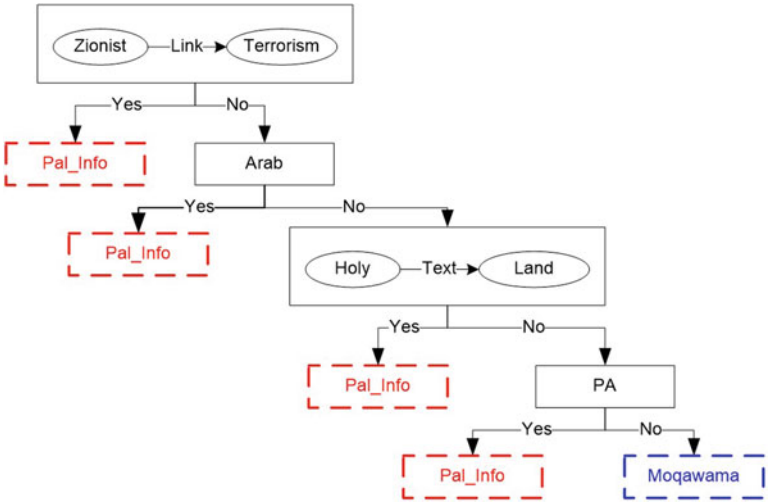


Fig. 4 C4.5 decision tree for classification of web pages in English [14]

"Zionist Terrorism" and the expression "Holy Land" in the document text). This simple decision tree can be interpreted as follows: if at least one of these four terms appears in an English document coming from one of these two Web sites, it can be safely labeled as "Hamas Propaganda." On the other hand, a document that contains none of these terms should be labeled as "Hezbollah Propaganda." The induced model indicates that, in contrast to Hamas, Hezbollah usually presents itself as an organization opposing "Israel" rather than "Zionist Terrorism" and their militant narrative, at least in English, does not emphasize the concept of "Holy Land."

2.4 Using Hybridized Term-Weighting Method for Terrorist Content Detection

Sabbah et al. [16] utilize a hybridized feature selection method based on basic term-weighting techniques for detection of terrorist-generated textual content in Arabic. In their work, they use the following popular weighting schemes for measuring the importance of a given term:

- Term Frequency (TF): Number of term occurrences in a given document.
- Document Frequency (DF): Number of documents containing a given term.
- Term Frequency—Inverse Document Frequency (TF-IDF): Terms of a given in-document frequency are more important if they appear in fewer documents.
- Glasgow: Terms of a given in-document frequency are more important if they appear in shorter documents.
- Entropy: Terms of a given in-document frequency are more important if they appear in fewer documents and have a higher entropy of in-document frequency over the entire corpus.

The subsets of top K terms selected by each one of the above schemes are combined using two different hybridization functions: UNION and Symmetric Difference. The number of subsets ("views") to be combined at a time varies between one and five. The resulting term vectors representing each document are used with the following popular classification techniques: Support Vector Machine (SVM), K Nearest Neighbor (KNN), Decision Trees (DT), Naive Bayes (NB), and Extreme Learning Machine (ELM).

The experiments were conducted on a balanced dataset of 500 Arabic documents downloaded from the Dark Web Forum Portal [17] and labeled as "dark" and 500 documents collected from multiple websites in Arabic and labeled as "non-dark." The maximum F-measure and accuracy value of 96.00% was reached with the SVM classifier using a UNION-based hybridized set of 1040 predictive features. The paper provides no information on the terms having the highest discriminative power according to the various weighting schemes.

2.5 Using Linguistic Markers for Detecting Violent Online Content

Johansson et al. [18] utilize a predefined set of *linguistic markers* to discover violent textual content in social media. These markers are aimed at detecting the following categories of warning behaviors:

- Leakage—communication of intent to do harm to a specific target (e.g., a government building, a house of prayer, etc.).
- Fixation—an increasing pathological preoccupation with a person (e.g., a public figure) or a cause (e.g., white supremacy or ISIS (Islamic State in Iraq and Syria)).

- Identification—a desire to be a "pseudo-commando," i.e., have a warrior or a martyr mentality.

A manual list of keywords and keyphrases is defined for each category. Since such a list may not cover the entire vocabulary of terms used by the online users, it is extended by a distributional semantic lexicon to ensure the lists of terms are exhaustive and up-to-date. Each addition to the keyword lists has to be confirmed by a human operator.

A monitoring tool performs a keyword search in the incoming stream of social media data for occurrences of the linguistic markers. For each marker, the monitoring tool outputs a list of Uniform Resource Identifiers (URIs) and the frequency of occurrence of the marker in the document. Then an automated system can specify the amount of markers that have to be triggered in order for an URI to be shown as potentially relevant to the user of the system. Sentiment analysis with respect to specific targets of interest may serve as an additional content filter.

In an initial three-day monitoring experiment, the authors detected 130 URIs that contained keywords matching both violence (intent) and at least one of bomb (intent) or weapons (intent). Applying an extra filter to these results so that they also should contain a negative sentiment against the Jews, only four URIs were left. Those originated from the website of a Neoconservative Right magazine, a Christian blog, and two anti-Jewish blogs.

3 Detecting Propaganda Campaigns on Social Media Platforms

3.1 Using Targeted Advertising on Facebook as a Propaganda Tool

Ribeiro et al. [19] explored the effectiveness of socially divisive political ads (e.g., discussing immigration, racial bias in policing, etc.), which were posted by a Russian propaganda group named Intelligence Research Agency (IRA) prior to 2016 U.S. elections using Facebook's targeted advertising platform. Since targeted ads are not seen by non-targeted and non-vulnerable users, malicious ads are likely to go unreported and their effects undetected. The study [19] is based on an in-depth analysis of a publicly released dataset of 3517 Facebook advertisements from 2015, 2016, and 2017 that are linked to IRA.

The click-through rate (CTR) of IRA-sponsored ads, computed as a ratio between the number of clicks and the number of impressions received by an ad, was found incredibly high. The median CTR was 10.8% and 75% of the ads had a CTR higher than 5.6. The average CTR was 10.8%. This value is almost ten times higher than the average CTR of 0.90% computed for Facebook ads across all industries (see Fig. 5).

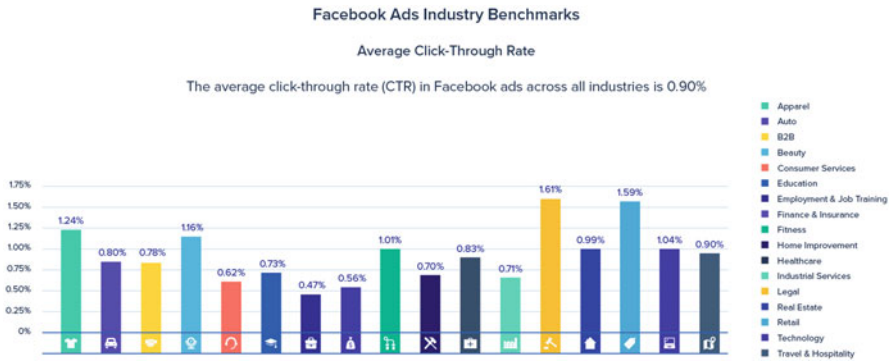


Fig. 5 Average click-through rate on Facebook in 2019. From https://instapage.com/blog/facebook-advertising-benchmarks

Moreover, many of IRA ads were severely divisive, and generated strongly varied opinions across the two ideological groups of liberals and conservatives. The divisiveness of an ad was calculated based on the differences in reactions of people with different ideological persuasions to the ad. These reactions were examined by three online surveys on a U.S. census-representative sample, which included 2886 respondents having a range of political views (40% liberal, 40% conservative, and 20% moderate or neutral). Each one of 485 high impact ads was shown to 15 different people who were asked about 10 different ads in one survey. Most these ads used attribute-based targeting, containing complex target formula that included interest and behavioral attributes suggested by Facebook. Specific divisiveness measures included (a) the likelihood of reporting the ad as inappropriate, (b) the strength of approving or disapproving the ad's content, and (c) the perception of false claims potentially presenting in the ad. Each one of these three measures was calculated both in terms of consistency within the same ideological group and in terms of differences between the respondents belonging to different ideological groups. In a significant percentage of the ads, liberals and conservatives completely disagreed with each other in their reporting and approval responses as well as in their perception of false claims.

3.2 Disinformation Campaigns on Twitter

Probably, the most cost-efficient way to distribute propaganda content on social media platforms, like Twitter, is to deploy a network of automated accounts, or bots. These are computer programs, which can post, tweet, and message on their own. Detection and subsequent blocking is not an issue for bots, since they can always be replaced at a negligible cost. Howard and Kollanyi [20] studied the use of political bots on Twitter during the UK referendum on EU membership. They

assumed that accounts posting more than 50 times a day have to be at least partially automated rather than being operated exclusively by human users. In fact, 32% of all Twitter traffic about Brexit was generated by less than 1% of all accounts, which tweeted 100 times or more per day. Moreover, the authors have concluded with a high degree of certainty that 7 of the 10 most active accounts generating traffic on StrongerIn-Brexit topics were actually bots.

The authors of [21] have analyzed 27,000 tweets posted during the 2016 US Presidential Campaign by 1000 accounts suspended by Twitter due to having ties with Russia's Internet Research Agency (IRA). These tweet posts are available from https://data.world/fivethirtyeight/russian-troll-tweets. Tables 1 and 2 show typical examples of such tweets related to Trump and posted before the Election Day in English and Russian, respectively.

According to the findings of [21], some of these Russian troll accounts disguise themselves as news portals by using the term "news" in their screen name and/or description. The resemblance of this deceitful tactics to Hamas naming their website "The Palestinian Information Center" is hard to ignore. After continuously monitoring the troll accounts data during the period of 21 months (from January 2016 to September 2017), they found several interesting differences between the Russian trolls and baseline (random) Twitter users:

- The Russian troll accounts use a smaller number of languages in their tweets (mostly, English and Russian) as opposed to random users.
- The trolls use less different clients for posting their tweets. The primary choice for most of them is a Web client rather than a mobile device.
- The self-reported location of Russian trolls (mostly, US, Russia, and Germany) is much more stable over time than the location reported by random users.
- Russian trolls use less images and video in their tweets, putting more emphasis on text.
- Russian trolls post more tweets containing at least one hashtag. Their top hashtags include mostly generic terms like #news, #politics, and #sports. Some hashtags such as #ISIS and #IslamKills represent dissemination of propaganda and/or controversial topics.
- Trolls' tweets include less *mentions*, which are associated with a smaller amount of Twitter users.
- Russian trolls include more URLs in their tweets compared to the baseline, probably to increase the credibility of their propaganda messages.
- Russian tweets sentiment tends to be more negative or neutral as opposed to mostly positive benchmark tweets. This confirms the trolls intention to increase the divisiveness of their audience.
- Using an LDA model, it was found that topics from Russian trolls often refer to specific events and political issues vs. more general issues discussed by benchmark tweets.
- Russian trolls change their screen names less frequently than benchmark users.

Table 1 Examples of IRA troll tweets in English[a]

Author	Content	Region	publish_date	post_type	account_type
BRIANWARNING	#TrumpBecause The Trump Train is unstoppable!	United States	04/08/15	Right	
BRIANWARNING	#TrumpBecause People we need a leader like @realDonaldTrump to pull out country out of this rut!	United States	04/08/15	Right	
BOOTH_PRINCE	I hate trump. No one loves him anyway. At least no sane person #IHate____Because	United States	02/07/16	RETWEET	Hashtager
BRIANWARNING	Thank You Mr.Trump! Always working, turning down millions and millions of dirty donations to expose the rats & lies.	United States	03/02/16	RETWEET	Right
BLMSOLDIER	The story that got Ted Cruz heated: 'Trump links Cruz's father to JFK assassin, channeling National Enquirer'	United States	03/05/16	RETWEET	Left
BOOTH_PRINCE	Mock the orangutans by showing them Trump memes #ThingsNotToDoAtTheZoo	United States	03/06/16	RETWEET	Hashtager
BLMSOLDIER	.@realDonaldTrump is a spiteful, petulant and gleeful mouthpiece for the lizard brain.	United States	04/05/16	RETWEET	Left
BOOTH_PRINCE	_I Will Tell My Grandchildren About The Time when Trump tried to make America great but unsuccessfully	United States	08/06/16	Hashtager	
BLMSOLDIER	Only 18 women work for the Trump campaign, and they make much less than his 52 male staffers	United States	09/06/16	RETWEET	Left
BLMSOLDIER	.@TimOBrien saw Trump's tax returns, and thinks you should, too	United States	12/05/16	RETWEET	Left

[a] Tweets referring to Trump during the 2016 Presidential elections campaign

Table 2 Examples of IRA troll tweets in Russian[a]

Author	Content	Region	publish_date	Translation
POLITICS_T0DAY	Гегемония США в цифрах и динамике: доктрина Трампа вполне рациональна.	United States	02/05/16	US hegemony in numbers and dynamics: Trump's doctrine is quite rational
POLITICS_T0DAY	«Трамп, Трамп, Трамп»: речь Клинтон о внешней политике свелась к критике...	United States	03/06/16	"Trump, Trump, Trump": Clinton's speech on foreign policy came down to criticism ...
POLITICS_T0DAY	«Обама» пишем, «Трамп» в уме Как США готовятся к борьбе за реальное мировое владычество.	United States	04/05/16	"Obama" we write, "Trump" we think. How is the US preparing to the struggle for a real control of the world
POLITICS_T0DAY	Трамп: криминальный Дон от политики рвется в Белый Дом.	United States	04/18/16	Trump: a criminal mafia boss in politics is rushing to the White House
POLITICS_T0DAY	Трамп: у США и России будут отличные отношения .	United States	04/27/16	Trump: the US and Russia will have excellent relationships
POLITICS_T0DAY	#Times: #Трамп взял террористов на ядерный прицел.	United States	04/29/16	Trump took terrorists into a nuclear sight
POLITICS_T0DAY	Берни Сандерс: Трампа стыдятся даже здравомыслящие республиканцы.	United States	05/04/16	Bernie Sanders: Even sane Republicans are ashamed of Trump
POLITICS_T0DAY	"У России с Клинтон вариантов нет": почему президент Трамп лучше .	United States	05/05/16	"Russia has no options with Clinton": why President Trump is better
POLITICS_T0DAY	Трамп пошел по стопам стендап-комика.	United States	09/04/16	Trump followed the footsteps of a stand-up comedian
POLITICS_T0DAY	Чего хочет Трамп и от чьего имени он говорит?	United States	10/05/16	What does Trump want and on whose behalf does he speak?

[a] Tweets referring to Trump during the 2016 Presidential elections campaign

- On average, Russian trolls accumulated four times more followers and friends than the benchmark users over the same data collection period, which probably indicates the trolls effort to increase their reachability.
- Unlike benchmark accounts, trolls tend to delete their tweets in batches.
- The trolls influence on normal Twitter users appears to be stronger than the normal users influence on each other, probably because the trolls were used to disseminate propaganda to normal Twitter users.

The overall conclusion of this study was that the trolls influence on normal Twitter users was not substantial compared to the other platforms, with the significant exception of news published by RT (Russia Today), a Russian international television network funded by the Russian government.

3.3 Fake News Detection

Fake news is intentionally false or misleading information presented as factual news. This kind of disinformation can be spread on social media quickly and efficiently via the users who tend to trust their online contacts more than the professional media outlets. Detecting false claims in news reports is a challenging task because human evaluators have shown only marginal improvements over random guesses [7]. Consequently, an automated fake news detection system would be very helpful for social media websites who currently rely on their users to report posted news stories, which they suspect to be significantly inaccurate or completely fake.

The authors of [8] design and evaluate a deep convolutional neural network (FNDNet) for fake news detection. They use the GloVe pre-trained word embedding model of 100 dimensions for representing the text of a potentially unreliable news article. The input word vectors are distributed across three parallel convolutional layers having 128 filters each. After hyperparameter optimization of their model, the authors have reached the classification accuracy of 98.36% on the Kaggle fake news dataset, which was collected during the time of the 2016 U.S. Presidential Election.

3.4 Troll Accounts Detection

Ghanem et al. [22] attempt to detect online troll accounts using textual features extracted from the tweets posted by each Twitter user. Their study is based on Russian troll accounts from the IRA dataset that use English as main language. They have also removed all non-English tweets from the data of those accounts. To represent the behavior of normal Twitter users, as opposed to trolls, they have collected a random sample of regular accounts having the following characteristics:

- The user declared region was US.
- The user chose English as the language of the Twitter interface.
- The user posted at least 5 tweets between 1st of August and 31 of December, 2016 assuming that Russian trolls were most active during that period.
- The account used hashtags related to the elections and their parties (e.g., #trump, #clinton, #election, #debate, #vote, etc.).

Using a topic modeling algorithm, the tweets content was associated with seven main themes such as *Police shootings*, *Islam and War*, etc. The theme-based features extracted from each tweet represented the following propaganda-related aspects of the posted content: emotions, sentiment, bad & sexual cues, stance cues, bias cues, linguistic categories, and morality. Finally, the users were modeled by the mean and the standard deviation of the above textual features calculated over their tweets on each theme. In addition, the following two sets of features were calculated over all tweets of each user (disregarding their topic): Native Language Identification (NLI) features and stylistic features.

The account classification performance was measured by $F1_{macro}$ value using 5-fold cross-validation. The best results were obtained with the logistic regression classifier. Though combining all proposed text-based features led to the highest F1 value of 0.94, a model based only on the Native Language Identification (NLI) features was nearly as accurate ($F1 = 0.91$), presumably because a large amount of IRA English tweets was written by native Russian speakers. Stylistic and theme-based features have provided the F1 values of 0.88 and lower.

3.5 Propaganda and Misinformation in Social Media

As indicated in [23], social media has been increasingly used for dissemination of misinformation, which is "false or inaccurate information that is deliberately created and is intentionally or unintentionally propagated." The categories of misinformation most frequently used for propaganda purposes include: intentionally spread misinformation, fake news, trolls, and hate speech. Misinformation detection methods are similar to other propaganda detection algorithms. Here are the main detection approaches presented in [23]:

- Content-based misinformation detection: usually based on supervised text categorization methods trained on collections of labeled content.
- Context-based misinformation detection: based on the assumption that misinformation is intentionally promoted by certain groups of accounts and thus has special posting patterns in terms of location and time.
- Propagation-based misinformation detection: detecting misinformation based on the information diffusion patterns in a social network.
- Early detection of misinformation: The earliness, or timeliness of a detection method refers to the need of detecting misinformation as early as possible, before it becomes widespread in the social media. This is known as a challenging issue

due to a lack of data and lack of labels. However, the waiting period can be shortened by utilizing structural information such as hashtags, web links and content similarity.

Finally, Wu et al. [23] indicate several directions for future research in countering the spread of misinformation such as detecting the most influential spreaders of misinformation, exploring the most effective spreading behaviors, and developing detection methods that are robust to adversarial attacks by sophisticated misinformation spreaders who are knowledgeable of the detection algorithms.

4 Conclusion and Future Research

In this survey chapter, we have provided an overview of state-of-the-art methods for automated detection of online propaganda content disseminated via social media outlets and traditional web platforms. Though a significant progress has been made in detecting textual propaganda in English, identification and understanding of non-English propaganda messages remains a major challenge due to a shortage of labeled data and limitations of current machine translation tools. For example, the automatic translation of most Russian language tweets appearing in Table 2 by a popular online translation service resulted in significant inaccuracies up to distortion of the tweets actual meaning. Another open challenge is detecting propaganda messages in multimedia content such as images, audio, and video.

References

1. Smith BL (2019) Propaganda. In: *Encyclopedia Britannica*. https://www.britannica.com/topic/propaganda. Cited 07 Sep 2019
2. Hirst KK (2019) Behistun Inscription: Darius's Message to the Persian Empire. In: *ThoughtCo*, http://thoughtco.com/behistun-inscription-dariuss-message-170214. Cited 07 Sep 2019
3. Wikisource contributors (2019) On running for the Consulship. In: *Wikisource*. https://en.wikisource.org/. Cited 13 October 2019
4. Cyber-terrorism (1997). Foreign Report, London, 25 September 1997.
5. Cashell B, Jackson W, Jickling M, et al (2004) The Economic Impact of Cyber-Attacks. *CRS Report for Congress*.
6. The 9/11 Commission Report, July 22, 2004.
7. Babaei M, Chakraborty A, Kulshrestha J, et al. (2019, January). Analyzing biases in perception of truth in news stories and their implications for fact checking. In Proceedings of the Conference on Fairness, Accountability, and Transparency (pp. 139–139).
8. Kaliyar RK, Goswami A, Narang P, & Sinha, S (2020). FNDNet–a deep convolutional neural network for fake news detection. Cognitive Systems Research, 61, 32–44.
9. MSNBC News Services(2006) Pentagon Surfing 5,000 Jihadist Web Sites. http://www.msnbc.msn.com/id/12634238/. Cited 22 March 2008
10. Harding B (2006) 29 Charged in Madrid Train Bombings. In: *Scotsman news*. http://news.scotsman.com/topics.cfm?tid=1094&id=556932006. Cited 29 Nov 2007

11. Lyall S (2006) London Bombers Tied to Internet, Not Al Qaeda, Newspaper Says. In: *New York Times*, April 11, 2006. http://www.nytimes.com/2006/04/11/world/europe/11london.html. Cited 29 Nov 2007
12. Gill P, Corner E, Conway M et al (2017) Terrorist Use of the Internet by the Numbers. Criminology & Public Policy 16: 99–117.
13. Alalouf N, Friedman M, Last M (2011) Automated Detection of Terrorist Web Documents using Fuzzy Lexicons with Fuzzy-based Clustering Methodology. In: Duman E, Atiya A (eds) Use of Risk Analysis in Computer-Aided Persuasion, NATO Science for Peace and Security Series - E: Human and Societal Dynamics, Volume 88, IOS Press, pp. 113–129.
14. Last M, Markov A, Kandel A (2008) Multi-lingual Detection of Web Terrorist Content. In: Chen H, Yang C (eds) *Intelligence and Security Informatics*, Springer Berlin/Heidelberg, Studies in Computational Intelligence, Vol. 135, pp. 79–96.
15. Schenker A, Bunke H, Last M, Kandel A (2005) *Graph-Theoretic Techniques for Web Content Mining*. World Scientific, Series in Machine Perception and Artificial Intelligence, Vol. 62, Singapore.
16. Sabbah T, Selamat A, Selamat M, et al (2016) Hybridized term-weighting method for Dark Web classification. Neurocomputing 173:3:1908–1926.
17. Chen H (2012) Dark Web Forum Portal. In: Dark Web, Integrated Series in Information Systems, vol 30., pp. 257–270. Springer, New York,
18. Johansson F, Kaati L, Sahlgren, M (2017). Detecting linguistic markers of violent extremism in online environments. In: Artificial Intelligence: Concepts, Methodologies, Tools, and Applications, pp. 2847–2863, IGI Global.
19. Ribeiro FN, Saha K, Babaei M et al (2019) On microtargeting socially divisive ads: A case study of Russia-linked ad campaigns on Facebook. In: Proceedings of the Conference on Fairness, Accountability, and Transparency, pp. 140–149, ACM.
20. Howard PN, Kollanyi B. (2016) Bots,# StrongerIn, and# Brexit: computational propaganda during the UK-EU referendum. Available at SSRN 2798311. 2016 Jun 20.
21. Zannettou S, Caulfield T, De Cristofaro E, et al (2019) Disinformation warfare: Understanding state-sponsored trolls on Twitter and their influence on the web. In: Companion Proceedings of The 2019 World Wide Web Conference 2019 May 13, pp. 218–226.
22. Ghanem B, Buscaldi D, Rosso P. (2019) TexTrolls: Identifying Russian Trolls on Twitter from a Textual Perspective. arXiv preprint arXiv:1910.01340. 2019 Oct 3.
23. Wu L, Morstatter F, Carley KM, Liu H. (2019) Misinformation in Social Media: Definition, Manipulation, and Detection. SIGKDD Explor. Newsl. 21, 2 (November 2019), 80–90. https://doi.org/10.1145/3373464.3373475

Interpretable Machine Learning for Financial Applications

Boris Kovalerchuk, Evgenii Vityaev, Alexander Demin, and Antoni Wilinski

October. This is one of the peculiarly dangerous months to speculate in stocks in. The others are July, January, September, April, November, May, March, June, December, August and February. Mark Twain, 1894

1 Introduction: Financial Tasks

Forecasting stock market, currency exchange rate, bank bankruptcies, understanding and managing financial risk, trading futures, credit rating, loan management, bank customer profiling, and money-laundering analyses are core financial tasks for machine learning [1, 26, 36, 55]. Stock market forecasting includes uncovering market trends, planning investment strategies, identifying the best time to purchase the stocks and what stocks to purchase. Financial institutions produce huge datasets, which build a foundation for approaching these enormously complex and dynamic problems with machine learning tools. Potential significant benefits of solving these problems have motivated extensive research for years.

The focus of this chapter is a growing area of interpretable machine learning for financial applications. Interpretability is required for ML models to be reliable and trustful for applications of machine learning models [72].

B. Kovalerchuk (✉)
Department of Computer Science, Central Washington University, Ellensburg, WA, USA
e-mail: borisk@cwu.edu

E. Vityaev
Sobolev Institute of Mathematics, Russian Academy of Sciences, Novosibirsk, Russia

A. Demin
Ershov Institute of Informatics, Russian Academy of Sciences, Novosibirsk, Russia

A. Wilinski
Department of Finance and Management, WSB University in Gdansk, Gdansk, Poland

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_32

Almost every computational method has been explored and used for *financial modeling for a long time*. We will name just a few studies: Monte-Carlo simulation of option pricing, finite-difference approach to interest rate derivatives, and fast Fourier transform for derivative pricing [6, 10, 14, 34]. Such developments augment the traditional technical analysis of stock market curves [54].

Machine learning (ML) as a process of *discovering useful patterns and correlations* has its own niche in financial modeling. Similarly, almost every ML method has been used in financial modeling. An incomplete list includes linear and nonlinear models, multi-layer neural networks, k-means and hierarchical clustering; k-nearest neighbors, decision tree analysis, regression, ARIMA, principal component analysis, Bayesian learning; and most recently deep learning for price formation in financial markets, directional movements prediction of S&P500 index, financial sentiment analysis and others [16, 32, 34, 38, 57–59].

Less traditional relational deterministic and probabilistic methods based on the first-order logic (FOL) are growing in many domains including finance [19, 20, 31, 39–42, 45, 50–54, 60–64, 66] due to the current surge of interest in interpretable machine learning and AI.

Bootstrapping and other evaluation techniques have been extensively used for improving ML results. Specifics of financial series analyses with ARIMA, neural networks, relational methods, support vector machines, and traditional technical analysis are discussed in [3, 40, 49, 54].

The naive approach to ML in finance assumes a cookbook instruction on "how to achieve the best result." Some publications continue to foster this unjustified belief. In fact, the only realistic approach proven to be successful is providing comparisons between different methods, showing their strengths and weaknesses relative to problem characteristics (problem ID) conceptually, and leaving for user the selection of the method, which likely fits the specific user's problem. This means a clear understanding that ML is still more art than hard science. Fortunately, now there is a growing number of books, which discuss the issues of matching tasks and methods in a regular way [21, 40]. For instance, understanding the power of first-order logic If-Then rules, over the decision trees, can significantly change and improve ML design. In comparison with other fields such as geology or medicine, where test of the forecast is expensive, difficult, and even dangerous, a trading forecast can be tested next hour or day, without cost and capital risk involved in real trading.

Attribute-based learning methods such as neural networks, the nearest neighbor's method, support vector machines, and decision trees dominate in ML financial applications. They are relatively simple, efficient, and handle noisy data, but are limited in using background knowledge and complex relations. The recent review [69] of ML models for financial market prediction revealed that the most commonly used models are support vector machines and neural networks. Unfortunately both are very limited in providing the interpretable models. That review references a single and quite old paper (2006), which deals with interpretability by using fuzzy rules showing that explainability of ML models in finance is in a very nascent stage.

Only a few recent papers deal with explainability of machine learning models in finance; one of them [71] explores mortgage default risk based on a linear logistic regression (Logit) and gradient tree boosting model (GTB). Both models are not directly interpretable. The authors do not build a surrogate interpretable model, which will approximate those models but focus on computing the importance of the features by estimating their Shapley values from given models. It is motivated by the abilities of Shapley values capturing impact of feature combinations stating that this method estimates key drivers of mortgage defaults, such as the loan-to-value ratio and current interest rate, which are in line with the findings of the economics and finance literature.

Unfortunately, interpretability approaches including the feature importance approach to interpret ML models have an important deficiency in general and for Shapley values specifically. How much the importance ranks of the features can explain the model, if the methods of getting these ranks are not interpretable themselves? In the case of the Shapley values, the computational formulas are not derived from the task at hand and are not justified by that task (e.g., mortgage risk), but use the analogy from the game theory [72]. Thus, there is a strong need and a vast area of opportunities for developing and applying more interpretable machine learning models in finance.

Relational machine learning techniques based on the first-order logic, which includes Inductive Logic Programming (ILP) [24, 40, 50–53] intend to overcome these limitations on *interpretability*. Previously these methods have been relatively computationally inefficient and had rather limited facilities for handling numerical data [12]. These methods are enhanced in both aspects [40] and are expanded to several domains from bioinformatics to robotics. We expect that the applications of these methods will grow due to the current demand for interpretable ML.

Various publications explored the use of ML methods like *hybrid architectures* of neural networks with genetic algorithms, chaos theory, and fuzzy logic in finance [70]. Also, many proprietary financial ML applications exist but rarely reported.

This chapter is organized as follows. Section 2 describes the methodologies of machine learning in finance, Sect. 3 presents ML models and practice in finance covering Potfolio managent with neural networks, Interpretable trading rules and relational ML, Relational machine learning for trading, Visual knowledge discovery for currency exchange trading, and Discovering money laundering and attribute-based relational machine learning. This chapter concludes with its summary and future studies.

Who can benefit from this chapter? The finance people already working in the field can see that the arsenal of ML methods in finance is expanded significantly with interpretable methods, such as relational methods based on first-order logic and Visual Machine Learning. The finance newcomers will get the same benefits.

Data science practitioners can see another important area of application for ML methods where they can work on. For developers of new ML methods, this is the area with new challenges to develop novel ML methods and tools.

2 Methodologies

2.1 Specifics of Machine Learning in Finance

Specifics of machine learning in finance are coming from the need to:

- Forecast multidimensional time series with high level of *noise*
- Accommodate specific *efficiency* criteria (e.g., the max of profit)
- Make coordinated *multiresolution* forecast (minutes, days, and so on)
- Incorporate *text* stream as input data (e.g., Enron case and September 11)
- *Explain the forecast* and the *forecasting model* ("black box" models have limited interest and future for significant investment decisions)
- Discover very *subtle patterns* with a *short lifetime*
- Incorporate the impact of market *players* on market regularities

The *efficient market theory/hypothesis* discourages attempt to discover long-term stable trading rules/regularities with significant profit. This theory is based on the idea that if such regularities exist, they would be discovered and used by most of the market players. This would make rules less profitable, and eventfully useless or even damaging. The market efficiency theory does not exclude that hidden *short-term local conditional regularities* may exist. These regularities cannot work "forever," they should be corrected *frequently*. It has been shown that the financial data are not random and that the efficient market hypothesis is merely a subset of a larger *chaotic market hypothesis* [23]. This hypothesis does not exclude successful short-term forecasting models for prediction of chaotic time series [13]. ML does not try to accept or reject the efficient market theory. It creates *tools* for discovering subtle short-term conditional patterns in financial data. Thus, retraining should be a permanent part of ML in finance and any claim that a silver bullet trading has been found should be treated similarly to claims that a *perpetuum mobile* has been discovered.

The impact of market players on market regularities stimulated a surge of attempts to use ideas of *statistical physics* in finance [11]. If an observer is a large marketplace player, then such observer can potentially change regularities of the marketplace dynamically. Attempts to forecast in such dynamic environment with thousands active agents lead to much more complex models than traditional ML models designed for. Therefore, such interactions are modeled using ideas from statistical physics. The *physics approach* in finance [35, 46] is also known as "econophysics" and "physics of finance." The major difference from the ML approach is that the physics approach is deeper integrated into the finance subject matter. ML approach covers empirical models and regularities derived directly from data and almost only from data with little domain knowledge explicitly involved. Historically, deep field-specific theories emerge after accumulating enough empirical regularities. We see that the future of ML in finance would be to generate more regularities that are empirical and combine them with domain knowledge via generic analytical ML approach [48]. The first attempts in this direction are presented in

[40] that exploit power of relational machine learning as a mechanism that permits to encode domain knowledge in the first-order logic language.

Data Selection and Forecast Horizon The selection of data for ML in finance is tightly connected to the selection of the target variable. There are several options for target variable *y:* $y = T(k + 1)$, $y = T(k + 2)$,. . . , $y = T(k + n)$, where $y = T(k + 1)$ represents forecast for the next time moment, and $y = T(k + n)$ represents forecast for n moments ahead. Selection of dataset $\boldsymbol{T}$ and its size for a specific desired forecast horizon n is a significant challenge. For stationary stochastic processes, the answer is well known – a better model can be built for longer training duration. For financial time series such as S&P500 index, this is not the case [47]. Longer training duration may produce many contradictory profit patterns that reflect bear and bull market periods. Models built using too short durations may suffer from overfitting, and hardly be applicable to the situations, where market is moving from the bull period to the bear period. The standard ML assumes that the model quality does not depend on *frequency* of its use. In finance frequency of trading is a model parameter, because *the model quality* includes both the accuracy of prediction and its profitability. The frequency of trading affects the profit, the trading rules, and strategy.

Measures of Success Traditionally the quality of forecasting models is measured by the standard deviation between the forecast and the actual values on training and testing data. For trading tasks two models with the same standard deviation can provide very different trading return [40]. A more specific success measure in financial ML is Average Monthly Excess Return (AMER) for industry i:

$$\mathrm{AMER}_j = R_{ij} - \beta_i R500_j - \Sigma_{j=1:12} \left(R_{ij} - \beta_i R500_j\right)/12$$

where R_{ij} is the average return for the S&P500 index in industry i and month j and $R500_j$ is the average return of the S&P500 in month j. The β_i values adjust the AMER for the index's sensitivity to the overall market. A second measure of return is Potential Trading Profits (PTP), which shows investor's trading profit versus the alternative investment based on the broader S&P500 index.

Quality of Patterns and Hypothesis Evaluation A typical approach is the testing of the null hypothesis H that pattern P is not statistically significant at level α. A meaningful statistical test requires that pattern parameters such as the month(s) of the year and the relevant sectoral index in a trading rule pattern P have been chosen *randomly* [30]. In many tasks, this is not the case.

Greenstone and Oyer argue that in the "summer swoon" trading rule, the parameters are not selected randomly. This means that rigorous test would require testing a different null hypothesis not only about one "significant" combination, but also about the *"family" of combinations.* A bootstrapping method was used to evaluate the statistical significance of such a hypothesis. Greenstone and Oyer [30] suggest a simple computational method – combining individual *t-test* results

by using the Bonferroni inequality that given any set of events $A_1, A_2, \ldots, A_n$, the probability of their union is smaller than or equal to the sum of their probabilities:

$$P\,(A_1\ \&\ A_2\ \&.\ldots\&\ A_k) \leq \Sigma_{i=1:k}\ P\,(A_i)\,.$$

2.2 Aspects of ML Methodology in Finance

ML in finance typically follows a set of general for any ML task steps such as problem understanding, data collection and refining, building a model, model evaluation, and deployment. An important step is adding expert-based rules in ML loop when dealing with absent or insufficient data. "Expert mining" is a valuable additional source of regularities. However, in finance, expert-based learning systems respond slowly to the market changes [18]. A technique for efficiently mining regularities from an *expert's perspective* has been offered in [40]. Such techniques need to be integrated into financial ML loop similar to what was done for medical ML applications [41].

Attribute-Based and Relational Methodologies Several parameters characterize machine-learning methodologies for financial forecasting. Data categories and mathematical algorithms are the most important among them. The first data type is represented by *attributes* of objects, that is each object x is given by a set of values $A_1(x), A_2(x),\ldots, A_n(x)$. The common ML methodology assumes this type of data, known as an *attribute-based* or *attribute-value methodology*. It covers a wide range of statistical and connectionist (neural network) methods.

The *relational data type* is a second type, where objects are represented by their relations with other objects, for instance, $x > y$, $y < z$, $x > z$. Here, we may not know that $x = 3$, $y = 1$, and $z = 2$. Thus, attributes of objects are not known, but their relations are known. Objects may have different attributes (e.g., $x = 5$, $y = 2$, and $z = 4$), but still have the same relations. The *relational methodology* is based on such a relational data type.

Another data characteristic, important for financial modeling methodology, is an actual *set of attributes* involved. A fundamental analysis uses all available attributes, but technical analysis uses only time series such as stock price, and parameters derived from it. Most popular time series are index value at open, index value at close, highest index value, lowest index value, and trading volume and lagged returns from the time series of interest. Fundamental factors include the price of gold, retail sales index, industrial production indices, and foreign currency exchange rates. Technical factors include variables, which are derived from time series, such as moving averages. The next characteristic is a form of the relationship between objects. Many ML methods assume its *functional form*, e.g., linearity of the border that discriminates between two classes, which is often hard to justify. Relational ML

does not assume a functional form but *learns symbolic relations* on numerical data of financial data.

Attribute-Based Relational Methodologies Below we discuss a combination of attribute-based and relational methodologies to mitigate their difficulties. Historically, relational ML was associated with *Inductive Logic Programming* (ILP), which is a deterministic technique in its purest form. The typical claim about relational ML is that it cannot handle large data assuming that input data are relations, which take more space than individual attributes. Computing relations from attribute-based data on demand resolves this issue. For instance, to explore a relation, Stock(t) > Stock($t + k$) for k days ahead we can compute it for every pair of stock data as needed. Multiple studies had shown that relational ML is most suitable for applications, where *structure can be extracted from the instances,* while Graph Neural Networks attempt to capture relations in a less interpretable way.

Problem ID and Method Profile Selection of a method for discovering regularities in financial time series is a very complex task. Uncertainty of problem descriptions, and method capabilities are among the most obvious difficulties in this process. To deal with this issue, a unified vocabulary, and a framework for matching the problems and methods have been proposed in [21]. A problem is described using a set of *desirable values* (problem ID profile) and a method is described using its *capabilities* in the same terms. Use of unified terms (*dimensions*) for problems and methods enhances the capabilities of comparing alternative methods. Introducing dimensions also accelerates their clarification. Next, users should not be forced to spend time determining a method's capabilities (values of dimensions for the method). This is a task for developers, but users should be able to identify desirable values of dimensions, using natural language terms [21].

In Sect. 3.3 an original relational machine learning method in presented with its capabilities (dimensions) relative to the desirable values in the problem ID profile. These capabilities are illustrated by rule (3) in Sect. 3.3.

Relational Machine Learning in Finance Decision trees are very popular in ML applications. They provide human readable, consistent rules, but discovering small trees for complex problems can be a significant challenge in finance [40]. In addition, DT rules fail to compare two attribute values, while it is possible with relational methods. Several publications strengthened that relational ML area is moving toward probabilistic first-order rules to avoid the limitations of deterministic systems. Relational methods in finance such as Machine Method for Discovering Regularities (MMDR) [40] are equipped with probabilistic mechanism, which is necessary for time series with high level of noise. MMDR is well suited to financial applications given its ability to handle numerical data with high levels of noise [18]. In computational experiments, trading strategies developed based on MMDR consistently outperform trading strategies developed based on other ML methods, and the buy-and-hold strategy.

3 ML Models and Practice in Finance

Prediction tasks in finance typically are posed as: (1) *straight prediction* of the market numeric characteristic, e.g., stock return or exchange rate, and (2) the prediction whether the market characteristic will *increase or decrease* no less than some threshold. Thus, the difference between ML methods for (1) or (2) can be less obvious, because (2) may require numeric forecast. Another type of task is assessment of investing risk [8]. It used a DT technique C5.0 and neural networks to a dataset of 27 variables for 52 countries whose investing risk category was assessed in a Wall Street Journal survey of international experts.

3.1 Portfolio Management and Neural Networks

Historically, the neural network most used by financial institutions was a multilayer perceptron (MLP) with a single hidden layer of nodes for time series prediction. The peak of such research activities was in mid-1990s [2, 27], which covered MLP and recurrent neural networks. Other neural networks used in prediction are time delay networks, Elman networks, Jordan networks, GMDH, and multi-recurrent networks [28]. Later we review the most recent work on deep neural networks (DNN).

Below we present typical steps of *portfolio management* using the neural network forecast of return values.

1. Collect 30–40 historical fundamental and technical factors for stock S_1, say for 10–20 years.
2. Build a neural network NN_1 for predicting the return values for stock S_1.
3. Repeat steps 1 and 2 for every stock S_i that is monitored by the investor. For example, 3000 stocks are monitored and 3000 networks, NN_i are generated.
4. Forecast stock return $S_i(t + k)$for each stock *i and k* days ahead (say a week, 7 days) by computing $NN_i(S_i(t)) = S(t + k)$.
5. Select n highest $S_i(t + k)$values of predicted stock return.
6. Compute a total forecasted return of selected stocks, T and compute $S_i(t + k)/T$. Invest to each stock proportionally to $S_i(t + k)/T$.
7. Recompute NN_i model for each stock i every k days adding new arrived data to the training set. Repeat all steps for the next portfolio adjustment.

These steps show why neural networks became so popular in finance. Potentially all steps above can be done automatically including the actual investment. Even institutional investors may have no resources to analyze manually the 3000 stocks and their 3000 neural networks every week. If investment decisions are made more often, say every day, then the motivation to use neural networks with their high adaptability is even more evident.

This consideration shows challenges of ML in finance – the need to build models that can be very quickly evaluated in both accuracy and *interpretability*. Because NN

are difficult to interpret even without time limitation steps 1–6 have been adjusted by adding more steps after step 3 that include *extracting interpretable rules* from the trained neural networks, and improving prediction accuracy using rules, e.g., [28] and more recently for DNN.

It is likely that extracting rules from the neural network is a *temporary solution*. It would be better to extract rules directly from data without introducing neural network artifacts to rules and potentially overlooking some better rules because of this. A growing number of computational experiments support this claim, e.g., [40] on experiments with S&P500, where first-order rules built directly from data outperformed backpropagation neural networks.

The logic of using ML in trading futures is like portfolio management. The most significant difference is that it is possible to substitute the numeric forecast of actual return for the categorical forecast, will it be profitable to buy or sell the stock at a price $S(t)$ on date t. This corresponds to long and short terms used in the stock market, where *Long* stands for buying the stock and *Short* stands for sell the stock.

Recently, due to the success of Deep Learning methods in various fields, the interest in using neural networks in financial forecasting problems has returned. Researchers actively use new neural network architectures and learning algorithms such as Deep Neural Network Classifier (DNNC), Long-Short Term Memory (LSTM), Gated Recurring Unit (GRU), and Convolutional Neural Networks (CNN) to solve financial tasks, including in conjunction with reinforcement learning to develop optimal solutions for the purchase and sale of assets [4, 16, 17, 37, 56]. However, in general, analyzing the results, we can conclude that the use of Deep Neural Networks (DNN) does not provide an obvious advantage compared to the simpler neural network models.

3.2 Interpretable Trading Rules and Relational ML

Comparison of Approaches Below we present the categories of rules, which can be discovered by different techniques. *Categorical rules* predict categorical attributes, such as increase/decrease and buy/sell. A typical example of a *monadic categorical rule* is the following rule:

$$\text{If } S_i(t) < \text{Value}_1 \text{ and } S_i(t-2) < \text{Value}_2 \text{ then } S_i(t+1) \text{ will increase.}$$

Here, $S_i(t)$ is a continuous variable, e.g., stock price at the moment t. If $S_i(t)$ *is* a discrete variable, then Value_1 and Value_2 are taken from m discrete values. This rule is called monadic, because it compared a *single attribute* value with a *constant*. Such rules can be discovered from trained decision trees by tracing their branches to the terminal nodes. Unfortunately, decision trees produce only such rules.

The technical analysis rule below is a *relational categorical rule*, because it *compares* values of 5- and 15-day moving averages (ME_5 and ME_{15}) and derivatives of moving averages for 10 and 30 days (DerivativeME_{10}, DerivativeME_{30}):

$$\text{If } \mathrm{ME}_5(t) = \mathrm{ME}_{15}(t) \ \&\ \text{Derivative } \mathrm{ME}_{10}(t) > 0 \ \&\ \text{Derivative } \mathrm{ME}_{30}(t) > 0$$
$$\text{then Buy stock at moment } (t+1).$$

This rule can be read as: If moving averages for 5 and 15 days are equal and derivatives for moving averages for 10 and 30 days are positive, then buy stock on the next day. Thus, classical for stock market technical analysis is superior to decision trees. This rule is written in a *first-order logic* form. Typically, technical analysis rules are not discovered in this form, but the relational ML technique does.

Classical categorical rules assume crisp relations such as $S_i(t) < \mathrm{Value}_1$ and $\mathrm{ME}_5(t) = \mathrm{ME}_{15}(t)$. More realistic is to assume that $\mathrm{ME}_5(t)$ and $\mathrm{ME}_{15}(t)$ are equal only approximately and Value_1 is not exact. Fuzzy logic and rough sets rules are used in finance to work with *"soft"* relations [9, 40]. The logic of using *"soft" trading rules* includes the conversion of time series to soft objects, discovering a temporal "soft" rule from stock market data, discovering a temporal "soft" rule from experts ("expert mining"), testing consistency of expert rules and rules extracted from data, and finally using rules for forecasting and trading.

Unlike neural networks, below we use a probabilistic logic network (PLN) approach, which is related to Probabilistic logic network [29]. The main distinct characteristics of PNL are as follows:

1. It uses first-order logic to record patterns, which makes it quite capable to discover a wide range of patterns.
2. These patterns are readable and understandable and belong to the field of Explainable Artificial Intelligence [22].
3. The inductive-statistical inference (I-S inference) is used to derive the predictions.
4. The disadvantage of I-S inference is its statistical ambiguity, which was resolved recently [60–62].

In PLN we define the maximally specific rules, the I-S derivation by which is logically consistent [62]. Inductive inference of rules is carried out by a special semantic probabilistic inference, during which the conditional probability of rules strictly grows, which distinguishes it from probabilistic logical programming. The most specific rules are obtained in the training process, which uses semantic probabilistic inference, which is implemented in the Discovery software system [63–66].

3.3 Relational Machine Learning for Trading

This section presents the application of the first-order logic relational approach to develop a trading system for the S&P500 index.

Let $c(t_1), \ldots, c(t_n)$ be values of S&P500 close at time moments t_1, t_2, .., t_n. The goal is to predict the direction of S&P500 (up or down) 5 days ahead. Thus, it is to predict the truth of the predicates $c(t_1) < c(t_1 + 5)$ and $c(t_1) > c(t_1 + 5)$.

First, we formulate the hypotheses to be tested on time series. The experience of classical technical analysis shows that before changing the direction of their movement, prices form the so-called figures of technical analysis [54]. Based on this idea, a class of hypotheses and predicates were developed. The following predicates were used for this:

Predicate $c(t_i) < c(t_j)$, which compares the time series value at t_i and t_j;

Predicate $\text{ext}(t_i) = \delta_i$, where δ_i is -1 or 1 and $\text{ext}(t_i) = -1$ means a local minimum of $c(t)$ is at time t_i.

$$(\text{ext}\,(t_i) = -1) \iff (c\,(t_i) < c\,(t_i - 1)) \;\; \& (c\,(t_i) < c\,(t_i + 1))\,,$$

$\text{ext}(t_i) = 1$ means a local maximum of $c(t)$ is at time t_i.

$$(\text{ext}\,(t_i) = 1) \iff (c\,(t_i - 1) < c\,(t_i)) \;\; \& \;\; (c\,(t_i + 1) < c\,(t_i))\,.$$

Below we use notation $\text{ext}(t_1, \ldots, t_n) = <\delta_1, \ldots, \delta_n>$, which is equivalent to $(\text{ext}(t_1) = \delta_1)$ & $\ldots$ & $(\text{ext}(t_n) = \delta_n)$.

The hypotheses will test whether a certain combination of points of local minima and maxima has been formed in the past of the time series. If such a combination is found, a forecast is made 5 days ahead. Here is an example of the rule:

$$\begin{aligned} \forall t_1 \exists (t_2, t_3, t_4, t_5, t_6) : (\text{ext}\,(t_1, t_2, t_3, t_4, t_5, t_6) =< -1, 1, -1, 1, -1, 1 >) \\ \& \;\; (c\,(t_1) < c\,(t_2)) \;\; \& \;\; (c\,(t_2) < c\,(t_4)) \;\; \& \;\; (c\,(t_3) < c\,(t_2)) \\ \& \;\; (c\,(t_5) < c\,(t_6)) \;\; \& \;\; (c\,(t_6) < c\,(t_4)) \rightarrow (c\,(t_1) > c\,(t_1 + 5)) \end{aligned} \tag{1}$$

Figure 1 shows a general view of the figure described by this rule. It says that if the time series formed a figure described by this rule, then the value of the series in 5 days will become less than the value of the series on the current day. This figure resembles the well-known figure "head-shoulders" of the technical analysis.

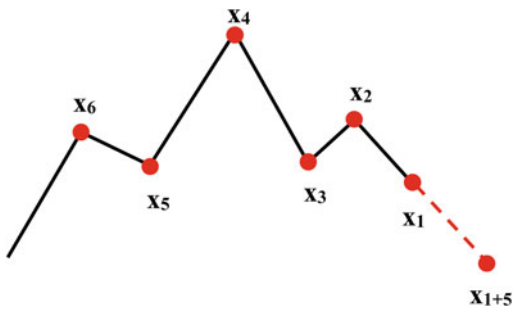


Fig. 1 Figure described by rule (1)

Since several different rules may work on the same trading day, we may have several forecasts, with different probability, predicting the direction of price movement. Moreover, since the rules are probabilistic, the forecasts may contradict each other. The next step is determining the trading strategy, which deals with contradictory predictions.

In this experiment, the final forecast is based on a comparison of the *maximum probabilities* of forecasts of an increase or decrease in price after 5 days. The *trading strategy* is to buy if the maximum probability that the price rises in 5 days is more than the maximum probability that it will fall and sell otherwise. A buy signal means to open a buy position for 5 days (buy and hold for 5 days, then close), if at the same time a sell position is already open, then it must be closed, If Signal(t) = 0, then do not trade on this day.

Testing We used for testing the method of moving control on the 9-year time interval. This interval includes 2065 trading days. A training window of size of 500 trading days and a testing interval of 100 days were used. Thus, for the entire testing period, the system passed 16 training and testing cycles, and the total testing interval was 1565 trading days. The quality of the trading system during testing was evaluated by modeling the real trading by assessing the potential profitability of the system. It requires ensuring that the system has a stable positive expected value:

$$P_{\text{Win}}\text{Trade}_{\text{Win}} + P_{\text{Loss}}\text{Trade}_{\text{Loss}} > 0, \tag{2}$$

where P_{Win} is the probability of gain, $\text{Trade}_{\text{Win}}$ is the average gain, P_{Loss} is the probability of loss, and $\text{Trade}_{\text{Loss}}$ is the average loss. The system trades with one unit of the contract and does not use stop-loss orders. While it is not optimal, it allows to fairly objectively evaluate the expected value (2).

Figure 2 shows the test results with the dynamics of capital growth over the entire test period (1565 days) where trading days are numbered from 1 to 1565. Table 1

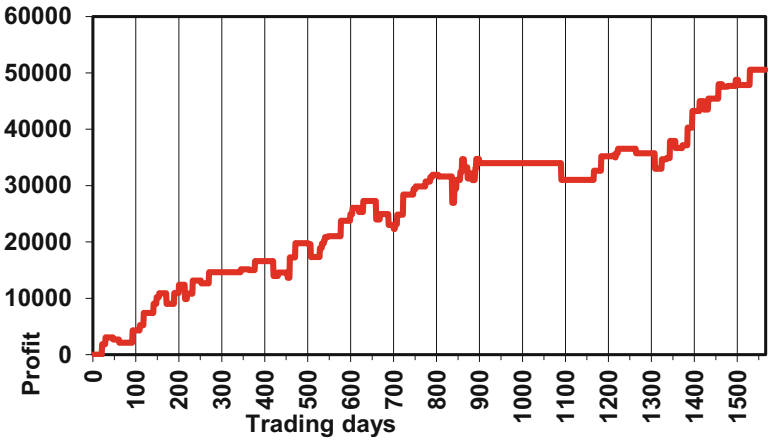


Fig. 2 Results of testing of the trading system

Table 1 Characteristics of the trading system

Indicator	Value
Profit	50,563
Maximum drawdown	−4960
Expected value	568
Percentage of profitable trades to all trades	69%
Annual rate of return in relation to the maximum drawdown of the account	238%

presents the indicators characterizing this trading system, where *Profit* is total profit that the trading system provided for the whole test period.

A *maximum drawdown* (MDD) is the maximum observed loss from a peak to a trough of a portfolio, before a new peak is attained. Maximum drawdown is an indicator of downside risk over a specified time [68],

$$\text{MDD} = (\text{Trough Value} - \text{Peak Value}) \,/\, (\text{Peak Value})\,;$$

The expected value (EV) is computed by summing up each of the possible outcomes multiplied by its likelihood.

Annual rate of return, in relation to the maximum drawdown of the account, characterizes the rate of return per unit of risk, where the value of the maximum drawdown of the account acts as a measure of risk. It is calculated by the formula:

$$\text{Annual Rate of return} = \frac{(\text{Profit})}{(\text{Maximum drawdown})} \cdot \frac{365}{(\text{Number of trading days})}.$$

As the test results show, this trading system has fairly stable positive expected values. Figure 2 shows that the system provides a steady growth of capital over the entire test period. The use of capital management methods and risk limitation would significantly improve this trading system; however, this is a topic for a separate study. Here is an example of the rules that were found during training:

$$\begin{aligned}\forall t_1 \exists t_2, t_3, t_4, t_5 : (\text{ext}\,(t_1, t_2, t_3, t_4, t_5) =< -1, 1, -1, -1, -1 >) \\ \&\ \ (c\,(t_5) < c\,(t_2))\ \ \&\ \ (c\,(t_4) < c\,(t_1))\ \ \&\ \ (c\,(t_4) < c\,(t_2))\ \ \&\ \ (c\,(t_3) < c\,(t_4)) \\ \&\ \ (c\,(t_1) < c\,(t_5)) \rightarrow (c\,(t_1) > c\,(t_1 + 5))\end{aligned} \tag{3}$$

This rule predicts a price reduction in 5 days. Figure 3 shows the general view of the figure described by this rule. The probability of this rule on the test set is 0.71.

For this rule, the results of a test set are shown in Fig. 4 and Table 2.

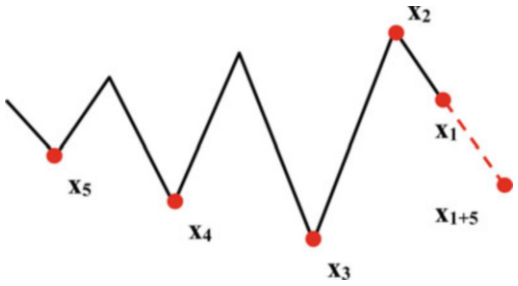


Fig. 3 Figure described by rule (3)

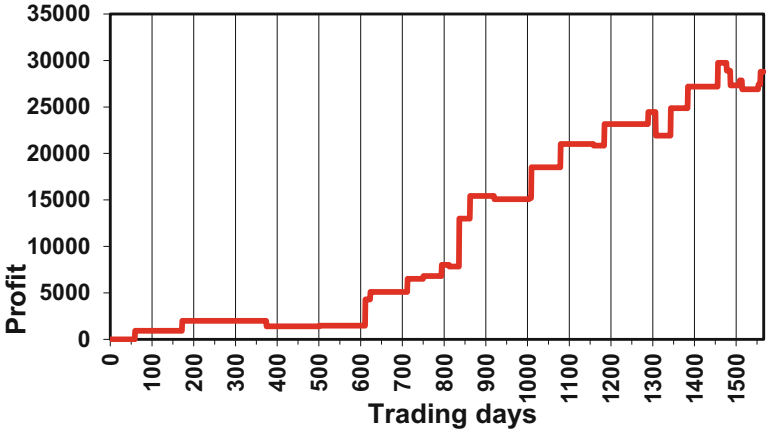


Fig. 4 Results of testing rule (3)

Table 2 Characteristics of rule (3)

Indicator	Value
Profit	28,786
Maximum drawdown	−2846
Expected value, see (3)	993
Percentage of profitable trades	72%
Annual rate of return in relation to the maximum drawdown of the account	236%

Comparison with Other Methods Comparison with other methods can be done in multiple ways. Below we start from desirable properties of the problem to be solved and the capabilities of the methods, which is conceptually outlined in Sect. 2.2 in terms of Problem ID and method profile.

To evaluate the efficiency of the Discovery system, we compared this system with the sliding linear regression method and neural networks. To compare the quality of various methods, it is necessary to choose a method for comparing the results of various methods. As applied to financial problems, such a comparison method can be a comparison of the financial performance indicators of the trading systems

based on these methods. Obviously, the method, whose forecast allows extracting more profit with less risk, has an advantage.

Sliding Linear Regression For an arbitrary function $c(t)$ represented by its values $c(t_k)$ at time moments t_k, a moving linear regression is constructed as a function $y(t) = a_n t + b_n$ using the last n values of the time series. Based on this function, the following value is predicted by the formula $y(t+1) = a_n(t+1) + b_n$. The trading strategy based on sliding linear regression is defined as follows:

$$\text{Signal}(t) = \begin{cases} 1, \text{if } y(t) < y(t+1) \\ -1, \text{if } y\,(t+1) < y(t) \\ 0, \text{otherwise} \end{cases}$$

Signal$(t) = 1$ means to open a buy position, if at the same time a sell position is already open, then it must be closed. Signal$(t) = -1$ means to open a sell position, if at the same time a buy position is already open, then it must be closed. Signal$(t) = 0$ means to keep open positions. The accuracy of the linear regression forecast largely depends on the choice of the size of the linear regression window. Linear regression was used for comparison, the window size of which was optimized, in order to obtain the best financial performance indicators of the trading system.

Neural Network For comparison, we used multilayer neural networks with direct connections, trained by back propagation of errors. A greater influence on the quality of forecasts of neural networks is provided by the method of presenting input information [25]. Here, there is a large correlation between successive values – the most probable course value at the next moment is equal to its previous value:

$$c(t) = c\,(t-1) + \Delta c(t) \approx c\,(t-1)\,.$$

At the same time, to improve the quality of training, one should strive for statistical independence of inputs, i.e., to the absence of such correlations. Therefore, the most significant values for prediction are not the values themselves, but their changes $\Delta c(t)$ as the most statistically independent values [25]. Based on these considerations, the initial series $c(t)$ were converted into a series of relative increments

$$d(t), d(t) = \Delta c(t)/c(t).$$

Here the input features of neural networks are k last values of the series: $d(t)$,$d(t-1)$,...,$d(t-k+1)$ and the output value is $(c(t+5) - c(t))/c(t)$, i.e., neural networks were trained to predict the relative change of c in 5 days.

Neural network training contains uncertainty associated with a random selection of initial weights. It leads to instability of neural networks for highly noisy financial time series. To increase the reliability of prediction, it is recommended to use a committee of neural networks [7]. We used a committee of 12 neural networks with different architectures and the following trading strategy:

Table 3 Comparison of methods

Indicator	Discovery system	Linear regression	Neural networks
Profit	50,563	27,805	33,117
Maximum drawdown	−4960	−19,059	−16,700
Expected value, see (3)	568	66	106
Percentage of profitable trades	69%	41%	57%
Annual rate of return in relation to the maximum drawdown of the account	238%	34%	46%

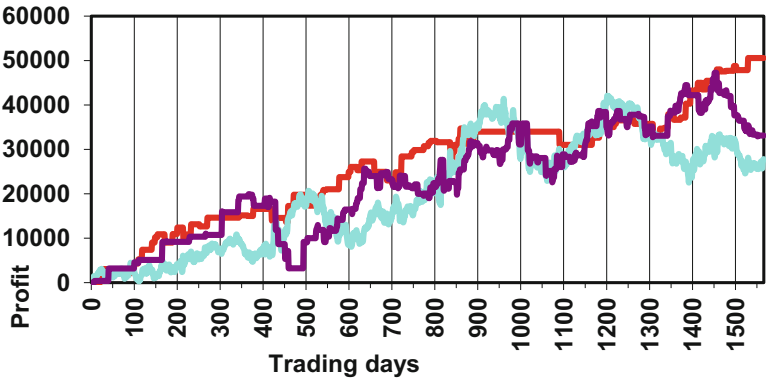


Fig. 5 Results of testing trading systems. Red – discovery system, blue –linear regression, violet – neural networks

$$\text{Signal}(t) = \begin{cases} 1, \text{if Out}(t) > 0 \\ -1, \text{if Out}(t) < 0 \\ 0, \text{otherwise} \end{cases},$$

where Out(t) is the output of the set of neural networks to time t, Signal(t) is a signal to trade on day t. Signal(t) $= 1$ means to open a position to buy for 5 days, and if the position is already opened it needs to be closed. Signal(t) $= -1$ means to open a position to sell for 5 days, and if the position is already opened it needs to be closed. Signal(t) $= 0$ means no trading on day t. Testing of this trading system was also carried out by moving control, with the same parameters.

Summary of the Comparison All methods were tested on the same time interval. Each trading system always entered the market with only one unit of the contract, and did not use stop-loss orders. Table 3 and Fig. 5 present the results of comparing the Discovery system with other methods. The graph in Fig. 5 shows the dynamics of capital growth over time for all three systems. Table 3 shows that the "Discovery" system surpasses other methods, in the percentage of correct forecasts and the indicators of the financial efficiency. Table 4 shows characteristics of the probabilistic logic network.

Table 4 Comparison of model quality and resources for probabilistic logic network (PLN) approach

Dimension	Desirable value for stock price forecast problem	Capability of PLN approach
Explainability	Desirable	Available
Ease to use logical relations	Desirable	Available
Ease to use numerical attributes	Required	Available
Tolerance for noise in data	Required	High
Tolerance for sparse data	Desirable	Available
Tolerance for complexity	Desirable	Available
Independence from experts	Desirable	Available

Comparing the profitability charts of trading systems, with the closing price chart of S&P500, shows that the trading system based on moving linear regression works well in areas with a noticeable trend. However, in places of a trend change and, especially, in the areas with very slight fluctuations of the price, it shows big losses. This is easily explained by the fact that linear regression at each moment of time tries to approximate the time series by a straight line, and this is acceptable only if there is a strong linear trend in the market.

Compared to linear regression, the neural network trading system works slightly better in the areas with very slight fluctuations of the price, but a trend change still has a catastrophic effect on the shape of the yield curve, causing large dips. Moreover, the careful analysis of the graphs shows that neural networks work well only in those areas where the overall price dynamics coincides with the price dynamics of the area in which they were trained. This is because neural networks try to approximate the entire training set, trying to minimize the average error, so they first find the most general patterns that are satisfied for most examples from the training set. Thus, neural networks basically "catch" the most typical dynamics of the time series and cease to work when the trend changes.

The relational system, in contrast to neural networks, can find the highly probable statistically significant patterns preceding a directional price movement. Most of these patterns continue to work successfully with trend changes.

3.4 Visual Knowledge Discovery for Currency Exchange Trading

This section presents a visual knowledge discovery approach to find an investment strategy in multidimensional space of financial time series [67]. Visualization based on the lossless Collocated Paired Coordinates (CPC) [43] plays an important role in this investment approach. The dedicated CPC subspaces constructed for EUR/USD foreign exchange market time series include characteristics of moving averages, differences between moving averages, changes in volume, and adjusted moving averages (Bollinger band).

In this study, the profit is analyzed in normalized units: price interest points (pip) and PPC (Profit per candle). A pip indicates the change in the exchange rate for a currency pair, where one pip is 0.0001 USD in the pair EURUSD, which is used as a measurement unit of change. Profit per candle (PPC) is the difference between the cumulative profit at the end and the start of the period divided by the number of candles in the period,

$$\mathrm{PPC} = \left(\mathrm{Profit}_{\mathrm{end}} - \mathrm{Profit}_{\mathrm{begin}}\right) / \left(i_{\mathrm{end}} - i_{\mathrm{begin}}\right),$$

where i_{end} and i_{begin} are the numbers of considered candles, e.g., one-hour candles.

Effective relations were found for one-hour EURUSD pair in 2-D and 3-D visualization spaces, which lead to a profitable investment decision (long, short position, or nothing) that can be used in algotrading mode.

Below we summarize steps of CPC:

1. Representing a normalized to [0,1] n-D point $\mathbf{x} = (x_1, x_2, \ldots, x_{n-1}, x_n)$, as a set of pairs $(x_1, x_2), \ldots, (x_i, x_{i+1}), \ldots, (x_{n-1}, x_n)$;
2. Drawing 2-D orthogonal Cartesian coordinates $(X_1, X_2), \ldots, (X_{n-1}, X_n)$ with all odd coordinates *collocated* on a single horizontal axis X and all even coordinates *collocated* on a single vertical axis Y;
3. Drawing each pair (x_i, x_{i+1}) in (X_i, X_{i+1});
4. Connecting pairs by arrows to form a graph $\mathbf{x}^*$: $(x_1, x_2) \rightarrow (x_3, x_4) \rightarrow \ldots \rightarrow (x_{n-1}, x_n)$.

This graph $\mathbf{x}^*$ represents n-D point $\mathbf{x}$ in 2-D losslessly, i.e., all values of $\mathbf{x}$ can be restored. Thus, this visualization is reversible representing all n-D data without loss of them. For the odd n the last pair can be (x_n, x_n) or $(x_n, 0)$. For 3-D visualization, the pairs are substituted by the triples $(x_1, x_2, x_3), \ldots, (x_i, x_{i+1}, x_{i+2}), \ldots, (x_{n-2}, x_{n-1}, x_n)$. For an arbitrary n, some coordinates are repeated to get n divisible by 3.

For time series, pairs of variables (x_i, x_{i+1}) can be sequential pairs of values at time t and $t + 1$. In this way, a 4-D point $\mathbf{x}$ is formed as $(v_t, y_t, v_{t+1}, y_{t+1})$, where v is volume and y is profit at two consecutive times t and $t + 1$. A 4-D point $(v_t, y_t, v_{t+1}, y_{t+1})$ is represented in 2-D as an arrow from 2-D point (v_t, y_t) to 2-D point (v_{t+1}, y_{t+1}), respectively. This simple graph fully represents 4-D data. It has a clear and simple meaning, for instance, the arrow going up and to the right indicates the growth in both profit and volume from time t to $t + 1$. To observe better the beginnings and ends of events, the time pairs starting from odd time t are visualized separately from time pairs starting from even time t.

Similarly, a 6-D point $\mathbf{x}$ can be $(v_t, d_{\mathrm{MA}t}, y_t, d_{\mathrm{MA},t+1}, v_{t+1}, y_{t+1})$, where d_{MA} is the difference between the moving averages for some windows. We represent it in 3-D as an arrow from 3-D point $(v_t, y_t, d_{\mathrm{MA}t})$ to 3-D point $(v_{t+1}, d_{\mathrm{MA},t+1}, y_{t+1})$. This simple graph fully represents 6-D data point with a clear meaning – the arrow going up and to the right indicates the growth in all three attributes: profit, d_{MA}, and volume from time t to $t + 1$.

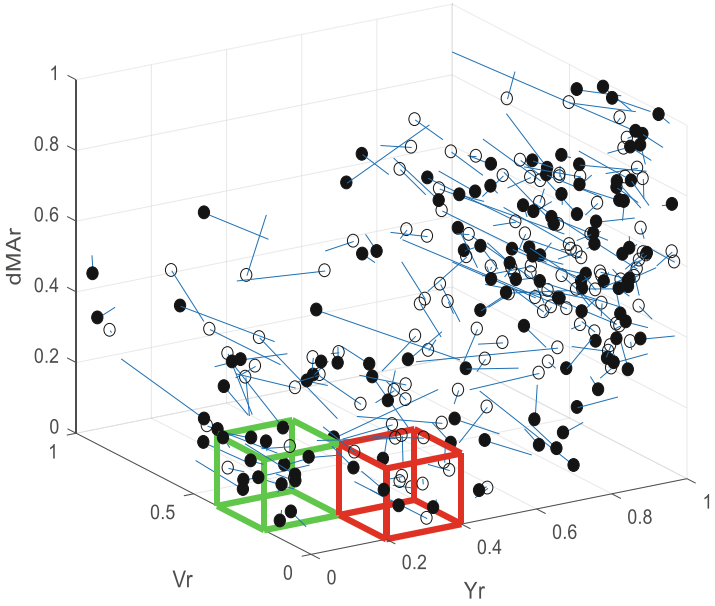


Fig. 6 Two discovered cubes in Y_r-$d_{\mathrm{MA}r}$-V_r space with the maximum asymmetry between long and short positions

To shorten notation for the spaces, we will use notation like (Y_r, V_r) instead of (Y_{rt}, V_{rt}, $Y_{r,t+1}$, $V_{r,t+1}$), and (Y_r, $d_{\mathrm{MA}r}$, V_r), instead of (Y_{rt}, V_{rt}, $d_{\mathrm{MA}rt}$, $Y_{r,t+1}$, $d_{\mathrm{MA}r,t+1}$, $V_{r,t+1}$), where index r stands for normalized profit, volume, and d_{MA}. Thus, 2-D and 3-D notations (Y_r, V_r) and (Y_r, $d_{\mathrm{MA}r}$, V_r) represent 4-D and 6-D spaces, respectively. In figures below, pins represent the arrows in the graphs, where circles indicate the beginnings of the arrows, with filled circles for the long position, and empty circles for the short positions.

The CPC visualization gives an idea of how to create an investment strategy – to learn and discover places in 2-D/3-D CPC representation where asymmetry between number of suggestions to open long positions (filled circles) and short positions (empty circles) is high. A rectangle or a square in 2D space and a cube or a cuboid in 3D space have been used in the experiments. These areas are changing and need to be learned and updated regularly.

The learning mode includes finding the optimal size rectangles and cuboids. Figure 6 shows the 2 colored cubes, with the largest asymmetry factor found by the learning algorithm. The proper positions can be opened, when subsequent events are located in the cubes. It provided the positive prediction with 0.619 accuracy for long positions, 0.686 for short positions, and a threshold $T_{\min} = 10$ on the number of pin circles required in the cube. The cumulative profit of this strategy for 5000 candles in learning period and 1700 candles during testing period is not too rewarding, because of small number of positive events, with a large period without trade. It indicates the need for a more dynamic strategy.

Strategy Based on Quality of Events in the Cubes The next approach uses the sum of returns Y_r accumulated in the learning period. A new hypothesis is that the more vertical arrows (pins) lead to higher profit. Consider a cube indexed by (k_1, k_2, k_3) in a 3D grid, where k_1, k_2 $k_3 = 1, 2, \ldots, K$. We are interested in maximum of criterion C_l for long positions and C_s for short positions:

$$C_{l\ (k1,k2,k3)} = \sum \left(Y_{r\ i(k1,k2,k3)} - Y_{r\ i-1(k1,k2,k3)}\right)$$

when $(Y_{r\,i} - Y_{r\,i-1}) > 0$ for all i that belong to a learning period and for all cubes (k_1, k_2, k_3) and

$$C_{s\ (k1,k2,k3)} = \sum \left(Y_{r\ i(k1,k2,k3)} - Y_{r\ i-1(k1,k2,k3)}\right)$$

when $(Y_{r\,i} - Y_{r\,i-1}) < 0$ for all i that belong to learning period and for all cubes (k_1, k_2, k_3).

Recall that the beginning of the arrow ($Y_{r\,i-1}$) belongs to (k_1, k_2, k_3)-cube, not its head. For each learning period, the sums $C_{l\,(k1,\,k2,\,k3)}$ and $C_{s\,(k1,\,k2,\,k3)}$ are computed in every (k_1, k_2, k_3)-cube. A corresponding investment strategy is – if C_l dominates, then open a long position, else open a short position:

$$C_{l\ (k1,k2,k3)} > C_{s\ (k1,k2,k3)} + d_c \text{ then open long position,}$$

$$C_{s\ (k1,k2,k3)} > C_{l\ (k1,k2,k3)} + d_c \text{ then open short position,}$$

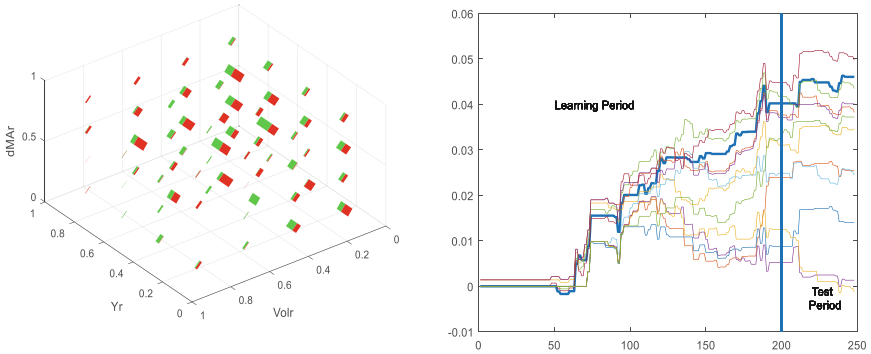
where d_c is additional difference for C_l and C_s to make the difference more distinct.

Figure 7a shows the bars, which represent the criteria C_l and C_s (by their length) in every cube (in grid $4 \times 4 \times 4$). The bars have different lengths. A visual strategy, based on this difference, is one of the bars is longer than the other one, then open a proper position. It was checked for different values of periods of learning and testing, i.e., bars have been generated for the test data, and compared with bars for the training data.

The thicker line in Fig. 7b is the best one, relative to the value of a linear combination of the cumulative profits, for the learning period, and Calmar ratio in the same learning period. For selecting the promising curve, the following criterion is used:

$$C = w_c \cdot \text{Calmar}_{\text{learn}} + w_p \cdot \text{profit}_{\text{learn}}$$

where w_c is a weight of Calmar component (in these experiments $w_c = 0.3$); w_p is a weight of profit component (in these experiments $w_p = 100$ for 500 hours of the learning period); $\text{Calmar}_{\text{learn}}$ is the Calmar ratio at the end of learning period; and $\text{profit}_{\text{learn}}$ is a cumulative profit at the end of the learning period measured as a change in EURUSD rate.



(a) (b)

Fig. 7 Bars in 3D after learning period which represent preferences to open long (green) or short (red) position and cumulative profits for different values of model parameters. (**a**) Bars in 3D space after learning period which represent preferences to open long (green) or short (red) position. (**b**) Cumulative profits for different values of parameters (f, s, k_b)

Here Calmar ratio, CR = (average of return over time Δt)/max drawdown over time Δt, which shows the quality of trading. In the literature the Calmar ratio of more than 5 is viewed as excellent, 2–5 as very good, and 1–2 as just good [67].

The curve with a significant profit and a small variance (captured by larger Calmar ratio for evaluating risk) is used in criterion C, as a measure of success of the algorithm, along with comparison of the result with typical benchmarks.

The main idea in constructing the criterion C is to balance risk and profit contributions with different pairs of weights (w_c, w_p).

Figure 7b shows that while the thicker line (best in criterion C) provides one of the top profits, it does not provide the best cumulative profits at the end of the learning period, due to the weighting nature of criterion C, which also takes into account the risk in the form of Calmar ratio.

Figures 8 and 9 show cumulative profits for different learning and testing periods as a function of 20 shifted training and testing datasets (cycles). First the best values of parameters (f, s, k_b) were found by optimizing them on training data, and then these values were used in the test period which follows the learning period. Figures 8 and 9 show very efficient results for 1 hour and 1 day taking into account the profit-risk relation.

Profit is represented by its pips value. These figures show the general direction of changes in cumulative profit, but y-axes have different scales and should not be compared directly. Practitioners consider the profit at the level of 10–20 pips per day as very high or even unrealistic to be stable [67]. The general conclusion is that the chosen space (Y_r, V_r, $d_{\mathrm{MA}r}$) is very efficient.

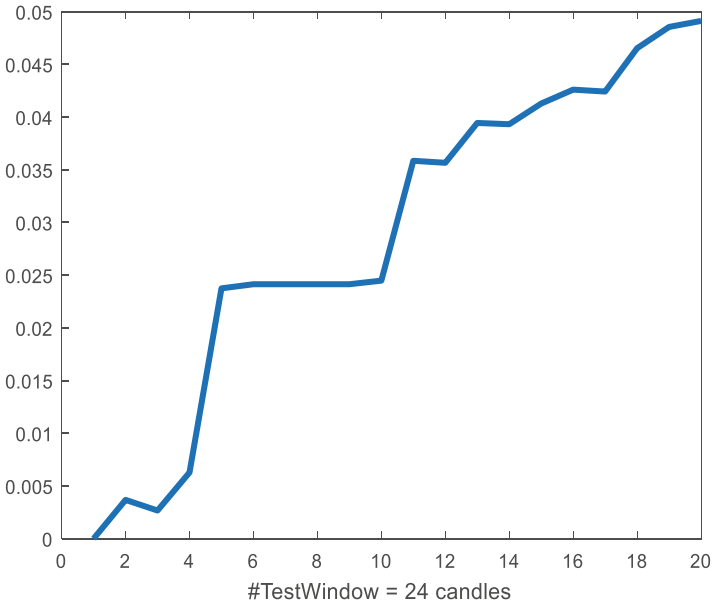


Fig. 8 Cumulative profit for main strategy with learning windows of 100 1 h-candles and test widows of 24 1 h-candles (Calmar ratio ~18, PPC = 0.87)

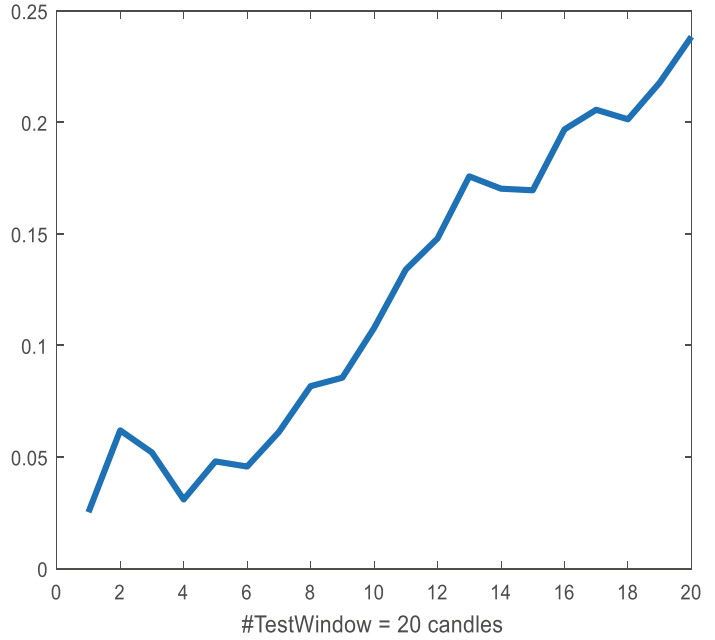


Fig. 9 Cumulative profit for the strategy in 1d EURUSD time series with 5-day testing windows and 30-day learning windows in 20 periods, Calmar ratio = 7.69; PPC = 23.83 pips; delta = 0.05

The learning period in Fig. 9 is about 1 month and the testing period is a 1-week period (5 trading days). These are very convenient for automatic and manual trading. For model detail, other experiments and positive comparison with Buy&Hold benchmark see [67]. The CPC concept can be applied more generally for financial time series. Besides CPC, the whole class of General Line Coordinates [43] opens multiple opportunities to represent *n*-D financial data visually and discovering patterns in these data. See a chapter on explainable machine learning and visual knowledge discovery in this handbook [44]. This section illustrates the potential of an emerging joint area of visual knowledge discovery and investment strategies, to boost the creativity of both scientists and practitioners.

3.5 Discovering Money Laundering and Attribute-Based Relational Machine Learning

Problem Statement Forensic accounting is a field, which deals with possible illegal and fraudulent financial transactions. One of the tasks in this field is the analysis of funding mechanisms for terrorism (Prentice 2002), where *clean money (e.g., charity money) and laundered money are both used* for a variety of activities including acquisition and production of weapons, and their precursors. In contrast, traditional illegal businesses and drug trafficking *make dirty money appear clean*.

The specific tasks in automated forensic accounting related to machine learning are the identification of suspicious and unusual electronic transactions and the reduction in the number of "false positive" suspicious transactions. Inexpensive, simple rule-based systems, customer profiling, statistical techniques, neural networks, fuzzy logic, and genetic algorithms are considered as appropriate tools.

There are many indicators of possible suspicious (abnormal) transactions in traditional illegal business. These include the following: (1) the use of several related and/or unrelated accounts before money is moved offshore, (2) a lack of account holder concern with commissions and fees, (3) correspondent banking transactions to offshore shell banks, (4) transferor insolvency after the transfer or insolvency at the time of transfer, (5) wire transfers to new places [15], (6) transactions without identifiable business purposes, and (7) transfers for less than reasonably equivalent value.

Some of these indicators can be easily implemented as simple flags in software. However, indicators such as wire transfers to new places produce a large number of "false positive" suspicious transactions. Thus, the goal is to develop more sophisticated mechanisms, based on interrelations of many indicators.

Machine learning can assist in discovering patterns of fraudulent activities, which are closely related to terrorism, such as transactions without identifiable business purposes. The problem is that often an individual transaction does not reveal that it has no identifiable business purpose, or that it was done for no reasonably equivalent value. Thus, machine-learning techniques can search for suspicious patterns in

the form of more complex combinations of transactions and other evidence using background knowledge. This means that the training data are formed not by transactions themselves but the combination of two, three, or more transactions.

This implies that the number of training objects exploded. The percentage of suspicion records in the set of all transactions is very small, but the percentage of suspicious combinations in the set of combinations is minuscule. This is a typical task of *discovering rare patterns*. Traditional machine learning methods and approaches are ill equipped to deal these such problems. Relational machine learning methods open new opportunities for solving these tasks by discovering "negated patterns" described below based on [42].

Approach and Method Consider a transactions dataset with attributes such as seller, buyer, item sold, item type, amount, cost, date, company name, type, and company type. We will denote each record in this dataset as (<S>, <B>, <I>), where <S>, <B>, and <I> are sets of attributes about the seller, buyer, and item, respectively. We may have two linked records $R_1 = (<S_1>, <B_1>, <I_1>)$ and $R_2 = (<S_2>, <B_2>, <I_2>)$, such that the first buyer B_1 is also a seller S_2, $B_1 = S_2$. It is also possible that the item sold in both records is the same $I_1 = I_2$. We create a new dataset of pairs of linked records $\{<R_1, R_2>\}$. ML methods will work in this dataset to discover suspicious records, if samples or definitions of normal and suspicious patterns provided. Below we list such patterns:

- A normal pattern (NP) – a Manufacturer Buys a Precursor & Sells the Result of manufacturing (MBPSR).
- A suspicious (abnormal) pattern (SP) – a Manufacturer Buys a Precursor and Sells the same Precursor (MBPSP).
- A suspicious pattern (SP) – a Trading Co. Buys a Precursor and Sells the same Precursor Cheaper (TBPSPC).
- A normal pattern (NP) – a Conglomerate Buys a Precursor & Sells the Result of manufacturing (CBPSR).

A machine learning algorithm A analyzes pairs of records $\{<R_1, R_2>\}$ with say 18 attributes total and can match a pair (#5,#6) with a normal pattern MBPSR, A(#5,#6) = MBPSR, while another pair (#1,#3) can be matched with a suspicious pattern, A(#1,#3) = MBPSP.

If the definitions of suspicious patterns are given, then finding suspicious records is a matter of a computationally efficient search in a database, which can be distributed. This is not the major challenge. The *automatic generation of patterns/hypotheses descriptions* is a major challenge. One can ask: "Why do we need to discover these definitions (rules) automatically?" A manual way can work if the number of types of suspicious patterns is small, and an expert is available. For multistage money-laundering transactions, this is difficult to accomplish manually. Creative criminals and terrorists permanently invent new and more sophisticated money-laundering schemes. There are no statistics for such new schemes, to learn as it is done in traditional machine learning approaches. An approach based on the idea of "*negated patterns*" can uncover such unique schemes. According to this

approach, *highly probable patterns* are discovered and then *negated.* It is assumed that a highly probable pattern should be *normal*. More formally, the *main hypothesis (MH)* of this approach is:

If Q is a highly probable pattern (>0.9) then Q constitutes a normal pattern and not(Q) can constitute a suspicious (abnormal) pattern

Below we outline an algorithm, based on this hypothesis, to find suspicious patterns. Computational experiments with two synthesized databases and few suspicious transaction schemes permitted us to discover transactions. The actual relational machine-learning algorithm used was algorithm MMDR (Machine Method for Discovery of Regularities). Previous research has shown that MMDR based on first-order logic and probabilistic semantic inference is computationally efficient, and complete for statistically significant patterns [40].

The algorithm for finding suspicious patterns based on the main hypothesis (MH) consists of four steps:

1. *Discover* patterns, compute probability of each pattern, select patterns with probabilities above a threshold, say 0.9. To be able to compute conditional probabilities patterns should have a rule form: IF A then B. Such patterns can be extracted using decision tree methods for relatively simple rules and using relational machine learning for discovering more complex rules. Neural Network (NN) and regression methods typically have no if-part. With additional effort, rules can be extracted from the NN and regression equations.
2. *Negate* patterns and compute *probability* of each negated pattern,
3. Find records database that satisfies negated patterns and analyze these records for possible *false alarm* (records maybe normal not suspicious).
4. Remove false alarm records and provide a detailed analysis of suspicious records.

The details can be found in [42].

4 Conclusion

To be successful, a machine learning project should be driven by the application needs, and results should be tested quickly. Financial applications provide a unique environment, where efficiency of the methods can be tested instantly, by not only using traditional training and testing data, but making real stock forecast and testing it the same day or week. This process can be repeated daily for several months collecting quality estimates. This chapter highlighted problems of ML in finance, and specific requirements for the ML methods, including in making interpretations, incorporating relations, and probabilistic learning.

The relational ML outlined in this chapter advances pattern discovery methods, which deal with complex numeric and non-numeric data, and involves structured objects, text, and data in a variety of discrete and continuous scales (nominal, order, absolute, and so on). This chapter shows the benefits of using such *interpretable*

methods for stock market forecast and forensic accounting, which includes uncovering money-laundering schemes. The technique combines first-order logic and probabilistic semantic inference. The approach has been illustrated with examples of discovery of suspicious patterns in forensic accounting, and stock market trading.

The success of machine learning exercises on automated trading systems has been reported in literature extensively, e.g., [33, 67]. For instance, machine-learning methods achieved better performance, than traditional statistical methods, in predicting bankruptcy and credit ratings, e.g., [5, 34].

The next direction is developing decision support software tools, specific for financial tasks, to adjust efficiently the financial ML models to a new data stream.

In the field of ML in finance, we expect an extensive growth of *hybrid methods*, which combine different analytical and visual models [43, 67] to provide a better performance than individual methods. In the integrative approach, individual models can serve as *trained artificial "experts."* Therefore, their combinations can be organized similar to a consultation of real *human experts*. Moreover, these artificial experts can be combined with real experts. These artificial experts can be built as autonomous *intelligent software agents.* Thus, "experts" to be combined can be machine learning models, real financial experts, trader and virtual *experts* (software intelligent agents), which run trading rules extracted from real experts. This requires "*expert mining*" for extracting knowledge from human experts to enhance virtual experts [41].

We expect that ML in finance will be shaped as a distinct field, which blends knowledge from finance and machine learning, similar to what we see now in bioinformatics, where integration of field specifics and machine learning is close to maturity. We also expect that the blending with ideas from the theory of dynamic systems, chaos theory, and physics of finance will deepen.

References

1. Alexander L, Das SR, Ives Z, Jagadish HV, Monteleoni C. Research challenges in financial data modeling and analysis. Big data. 2017 Sep 1;5(3):177–88.
2. Azoff, E., Neural networks time series forecasting of financial markets, Wiley, 1994.
3. Back, A., Weigend, A., A first application of independent component analysis to extracting structure from stock returns. Int. J. on Neural Systems, 8(4):473– 484, 1998.
4. Balaji, A. Ram D., Nair B., Applicability of Deep Learning Models for Stock Price Forecasting An Empirical Study on BANKEX Data, Procedia Computer Science, Volume 143, 2018, 947–953.
5. Barboza F, Kimura H, Altman E. Machine learning models and bankruptcy prediction. Expert Systems with Applications. 2017 Oct 15;83:405–17.
6. Baumann C, Elliott G, Burton S. Modeling customer satisfaction and loyalty: survey data versus data mining. Journal of services marketing. 2012 May 18;26(3):148–57.
7. Baxt W. G. Improving the accuracy of an artificial neural network using multiple differently trained networks, Neural Computation, 4(5), 772–780. 1992
8. Becerra-Fernandez, I., Zanakis, S. Walczak, S., Knowledge discovery techniques for predicting country investment risk, Computers and Industrial Engineering Vol. 43 , Issue 4:787 – 800, 2002.

9. Bojadziev G. Fuzzy logic for business, finance, and management. World Scientific; 2007.
10. Bormetti G, Callegaro G, Livieri G, Pallavicini A. A backward Monte Carlo approach to exotic option pricing. European J. of Applied Mathematics. 2018, 29(1):146–87.
11. Bouchaud, J., Potters, M., Theory of Financial Risks: From Statistical Physics to Risk Management, 2000, Cambridge Univ. Press, Cambridge, UK.
12. Bratko, I., Muggleton, S., Applications of Inductive Logic Programming. Communications of ACM, 38(11): 65–70, 1995.
13. Casdagli, M., Eubank S., (Eds). Nonlinear modeling and forecasting, Addison Wesley, 1992.
14. Cao J, Lian G, Roslan TR. Pricing variance swaps under stochastic volatility and stochastic interest rate. Applied Mathematics and Computation. 2016; 277:72–81.
15. Chabrow, E. Tracking the terrorists, Information week, Jan. 14, 2002.
16. Chen Y, He K, Tso GK. Forecasting crude oil prices: a deep learning based model. Procedia computer science. 2017, vol. 1;122:300–307.
17. Gonçalves R, Ribeiro VM, Pereira FL, Rocha AP. Deep Learning in Exchange Markets. Information Economics and Policy, Vol. 47, 2019, 38–51
18. Cowan, A., Book review: Data Mining in Finance, International journal of forecasting, Vol.18, Issue 1, 155–156, Jan-March 2002.
19. De Raedt L. Logical and relational learning. Springer, 2008.
20. De Raedt L., Kersting K, Natarajan S, Poole D. Statistical relational artificial intelligence: Logic, probability, and computation. Synthesis Lectures on Artificial Intelligence and Machine Learning. 2016 Mar 24;10(2):1–89.
21. Dhar, V., Stein, R., Intelligent decision support methods, Prentice Hall, 1997.
22. Došilović FK, Brčić M, Hlupić N. Explainable artificial intelligence: A survey. In: 2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO) 2018 (pp. 0210–0215). IEEE.
23. Drake, K., Kim Y., Abductive information modeling applied to financial time series forecasting, In: Nonlinear financial forecasting, Finance and Technology, 1997, 95–109.
24. Dzeroski S. , Relational data mining, in: Data mining and knowledge discovery Handbook. 2nd ed., O. Maimon, L. Rokach (Eds), pp. 887–911, Springer, 2010.
25. Ezhov, A., Shumski, S., Neurocomputing and its applications in economics and business, MFTI, Moscow,1998 (in Russian).
26. Fang B, Zhang P. Big data in finance. InBig data concepts, theories, and applications 2016, 391–412, Springer.
27. Freedman R., Klein R., Lederman J., Artificial intelligence in the capital markets, Irwin, Chicago, 1995.
28. Giles, G., Lawrence S., Tshoi, A. Rule inference for financial prediction using recurrent neural networks, In: Proc. of IEEE/IAAFE Conference on Computational Intelligence for financial Engineering, IEEE, NJ, 1997, 253–259.
29. Goertzel B, Iklé M, Goertzel IF, Heljakka A. Probabilistic logic networks: A comprehensive framework for uncertain inference. Springer, 2008.
30. Greenstone, M., Oyer, P., Are There Sectoral Anomalies Too? The Pitfalls of Unreported Multiple Hypothesis Testing and a Simple Solution, Review of Quantitative Finance and Accounting, 15, 2000: 37–55.
31. Gulwani S, Hernández-Orallo J, Kitzelmann E, Muggleton SH, Schmid U, Zorn B. Inductive programming meets the real world. Communications of the ACM. 2015;58(11):90–9.
32. Hajizadeh E, Ardakani HD, Shahrabi J. Application of data mining techniques in stock markets: A survey. Journal of Economics and International Finance. 2010 Jul 31;2(7):109–18.
33. Huang B, Huan Y, Xu LD, Zheng L, Zou Z. Automated trading systems statistical and machine learning methods and hardware implementation: a survey. Enterprise Information Systems. 2019 Jan 2;13(1):132–44.
34. Huang Z, Chen H, Hsu CJ, Chen WH, Wu S. Credit rating analysis with support vector machines and neural networks: a market comparative study. Decision support systems. 2004;37(4):543–58.

35. Ilinski, K., Physics of Finance: Gauge Modeling in Non-Equilibrium Pricing, Wiley, 2001
36. Jadhav I., He H., Jenkins K., An academic review: applications of data mining techniques in finance industry, International Journal of Soft Computing and Artificial Intelligence, ISSN: 2321-404X, Volume-4, Issue-1, May-2016
37. Jeong G., Kim H., Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning, Expert Systems with Applications, Vol. 117, 2019, 125–138.
38. Kingdon, J., Intelligent systems and financial forecasting. Springer, 1997.
39. Koller D, Friedman N, Džeroski S, Sutton C, McCallum A, Pfeffer A, Abbeel P, Wong MF, Heckerman D, Meek C, Neville J. Introduction to statistical relational learning. MIT press; 2007.
40. Kovalerchuk, B., Vityaev, E., Data Mining in Finance: Advances in Relational and Hybrid Methods, Kluwer, 2000.
41. Kovalerchuk, B., Vityaev E., Ruiz J.F., Consistent and Complete Data and "Expert Mining" in Medicine, In: Medical Data Mining and Knowledge Discovery, Springer,2001, 238–280.
42. Kovalerchuk B, Vityaev E, Holtfreter R. Correlation of complex evidence in forensic accounting using data mining. Journal of Forensic Accounting. 2007;8(1). Kovalerchuk B., Vityaev E., Symbolic Methodology for Numeric Data Mining. Intelligent Data Analysis. v.12(2), IOS Press, 2008, 165–188.
43. Kovalerchuk, B. Visual Knowledge Discovery and Machine Learning, 2018, Springer.
44. Kovalerchuk, B., Explainable Machine Learning Boosted by Visual Means, In: Handbook of Machine Learning for Data Science (this volume), 2020, Springer.
45. Lavrač N, Vavpetič A. Relational and semantic data mining. In: International Conference on Logic Programming and Nonmonotonic Reasoning 2015, 20–31, Springer.
46. Mantegna, R., Stanley, H., An Introduction to Econophysics: Correlations and Complexity in Finance, Cambridge Univ. Press, Cambridge, UK, 2000
47. Mehta, K., Bhattacharyya S., Adequacy of Training Data for Evolutionary Mining of Trading Rules, Decision support systems, 37(4):461–474, 2004
48. Mitchell, T., Machine learning. 1997, McGraw Hill.
49. Muller, K.-R., Smola, A., Rtsch, G., Schlkopf, B., Kohlmorgen, J., & Vapnik, V., 1997. Using support vector machines for time series prediction, In: Advances in Kernel Methods – Support Vector Learning, MIT Press, 1997.
50. Muggleton S., Scientific Knowledge Discovery Using Inductive Logic Programming. Communications of ACM, 42(11), 1999, 42–46.
51. Muggleton SH, Lin D, Tamaddoni-Nezhad A. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. Machine Learning. 2015 Jul 1;100(1):49–73.
52. Muggleton SH, Schmid U, Zeller C, Tamaddoni-Nezhad A, Besold T. Ultra-Strong Machine Learning: comprehensibility of programs learned with ILP. Machine Learning. 2018 Jul 1;107(7):1119–40.
53. Muggleton, S., Learning Structure and Parameters of Stochastic Logic Programs, 12th International Conference, ILP 2002, Sydney, Australia, July 9–11, 2002. Lecture Notes in Computer Science 2583 Springer 2003, 198–206.
54. Murphy JJ. Technical analysis of the financial markets: A comprehensive guide to trading methods and applications. Penguin; 1999.
55. Saxena A, Sharma N, Saxena K, Parikh SM. Financial Data Mining: Appropriate Selection of Tools, Techniques and Algorithms. In: Intern. Conf. on Smart Trends for Information Technology and Computer Communications 2017, 244–251. Springer.
56. Selvin S, Vinayakumar R, Gopalakrishnan EA, Menon VK, Soman KP. Stock price prediction using LSTM, RNN and CNN-sliding window model. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) 2017, 1643–1647, IEEE
57. Sirignano J, Cont R. Universal features of price formation in financial markets: perspectives from deep learning. Quantitative Finance. 2019 Sep 2;19(9):1449–59.

58. Sohangir S, Wang D, Pomeranets A, Khoshgoftaar TM. Big Data: Deep Learning for financial sentiment analysis. Journal of Big Data. 2018 Dec;5(1):3.
59. Vargas MR, De Lima BS, Evsukoff AG. Deep learning for stock market prediction from financial news articles. In: 2017 IEEE Intern. Conf. on Computational Intelligence and Virtual Environments for Measurement Systems and Applications, 2017, 60–65, IEEE.
60. Vityaev E. The logic of prediction. In: Mathematical logic in Asia 2006, 263–276, World Scientific Publ.
61. Vityaev E., Smerdov S. On the Problem of Prediction, K.E. Wolff et al. (Eds.): KONT/KPP 2007, LNAI 6581, 2011, 280–296, Springer.
62. Vityaev, E., Odintsov, S. How to predict consistently? In: Trends in Mathematics and Computational Intelligence. M. Cornejo (ed), 2019, 35–41, Springer.
63. Vityaev E., Kovalerchuk B. Ontological Data Mining. Uncertainty Modeling, Studies in Computational Intelligence 683, V. Kreinovich (ed.), Springer, 2017, pp. 277–292.
64. Vityaev E., Kovalerchuk B., Relational Methodology for Data Mining and Knowledge Discovery. Intelligent Data Analysis, v.12(2), 2008, 189–210, IOS Press.
65. Vityaev E., Kovalerchuk B., Empirical Theories Discovery based on the Measurement Theory. *Mind and Machine*, v.14, #4, 551–573, 2004.
66. Vityaev E., Perlovsky L., Kovalerchuk B., Speransky S., Probabilistic dynamic logic of cognition. Biologically Inspired Cognitive Architectures. 2013, 1;6:159–168.
67. Wilinski A, Kovalerchuk B. Visual knowledge discovery and machine learning for investment strategy. Cognitive Systems Research. 2017, 1;44:100–114.
68. Hayes A., Maximum Drawdown (MDD), Investopedia, 2020, https://www.investopedia.com/terms/m/maximum-drawdown-mdd.asp
69. Henrique BM, Sobreiro VA, Kimura H. Literature review: Machine learning techniques applied to financial market prediction. Expert Systems with Applications. 2019 Jun 15 ;124: 226–251.
70. Ang, K. & Quek, C. (2006). Stock trading using RSPOP: A novel rough set-based neuro-fuzzy approach. IEEE Transactions on Neural Networks, 17 (5), 1301–1315.
71. Bracke, P., Datta, A., Jung, C., & Sen, S. (2019). Machine learning explainability in finance: an ap-plication to default risk analysis, Bank of England, Paper No. 816, https://www.bankofengland.co.uk/-/media/boe/files/working-paper/2019/machine-learning-explainability-in-finance-an-application-to-default-risk-analysis.pdf
72. Kovalerchuk, B., Ahmad, M.A., Teredesai A., Survey of Explainable Machine Lear-ning with Visual and Granular Methods beyond Quasi-explanations, In: Interpretable Artificial Intelligence: A Perspective of Granular Computing (Eds. W. Pedrycz, S. M. Chen), Springer, 2021, 217–267; https://arxiv.org/abs/2009.10221

Predictive Analytics for Targeting Decisions

Jacob Zahavi

1 Introduction

Predictive Analytics (PA), one of the hottest topics in Data Science today, is concerned with predicting future events of individual entities (e.g., customers and prospects) based on past observations for which the response values are known. The event which we want to predict is the response variable, more commonly known as the dependent variable. The variables used to "explain" the event are the explanatory variables, also known as predictors, independent variables, attributes or, simply, variables. Data mining people refer to them as features.[1]

By and large, PA methods are divided into two main categories – classification and estimation. In classification, the objective is to classify observations to one of several predefined classes; in estimation, the objective is to predict the numeric value of the dependent variable.

The most common classification problems are the binary problems in which the objective is to classify future events into one of two possible classes Yes/No, or 0/1. The examples are many:

- Will the customer respond to an offer to purchase a product or a service?
- Will the customer churn and switch to a competitive company?
- Will the customer pay his/her debt in full?
- Will the customer click on an online ad?

[1] In the following we use the terms predictors, explanatory variables, variables, independent variables, attributes and features interchangeably

J. Zahavi (✉)
The Coller School of Management, Tel Aviv University, Tel Aviv, Israel
e-mail: jacobz@tauex.tau.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_33

- Will the customer recuperate from a given surgery?
- Will the customer file an insurance claim next year?

Multiple methods have been developed in recent years to entertain classification problems including, among the rest, logistic regression (Ben-Akiva and Lerman 1987), decision trees (Breiman et al. 1984), neural networks (Rumelhart et al. 1986), genetic algorithms (Goldberg 1989), Bayesian methods (Elkan 1997), SVM – support vector machines (Vapnik 1995), and others. The dependent variable in these methods is binary (0/1) and the independent variables are a collection of domain-related variables and customers' characteristics. Since it is not possible to predict future events for sure, all classification methods estimate the occurrence likelihood of the event. Depending upon the model used, the likelihood is expressed either by means of probability measures (such as in logistic regression), or some type of a ranking measures (such as in neural networks) – often the higher the measure, the higher the occurrence likelihood.

In estimation problems the dependent variable is continuous or countable, and the objective is to predict the value of this variable, usually its expected value, as a function of the collection of the explanatory variables. Also, here, the examples are many:

- How much money will the customer spend on purchasing from a catalog?
- How many items will the customer order from a catalog?
- How much money will the customer donate to a given charity?
- How many insurance claims will the customer file next year?
- How big will the insurance claim be?
- What is the LTV (lifetime value) of the customer?

The leading models for estimation problems consist of a variety of regression models, neural networks, decision trees, and other machine learning methods. A good reference on PA in general can be found in the book by Kelleher et al. (2015).

The major problem afflicting PA models in the world of big data, continuous as well as discrete, is the dimensionality issue which renders the model-building process very tedious and time-consuming. The most complex problem is feature selection – selecting the most influential predictors explaining the response from among a much larger set of potential predictors, which can reach hundreds, even thousands, of predictors. The objective is to choose the predictors which optimize a given goodness of fit function subject to constraints that renders stable models and avoid over-fitting.

A detailed discussion of some of the popular models used to address targeting decisions appears in Levin and Zahavi (2005). In this article, we provide an overall review of the process involved in building large-scale PA models in the world of big data with a focus on targeting decisions. Without loss of generality, we concentrate in this article on binary classification methods based on logistic regression. There are two conflicting concerns here: model accuracy represented by the bias caused by the difference between the "true" model parameters and the estimated model parameters (the specification error), and the sampling variance caused by the fact that the model parameters are estimated based on a random sample. The objective is to choose a model that both accurately capture the regularities in the learning dataset

(i.e., reduce bias) but also generalizes well to unseen data and new observations. Unfortunately, it may not be possible to reduce both the bias and variance errors. A bias-variance trade-off is therefore required to find the "best" set of predictors for a model.

2 PA – Predictive Analytics

Prediction problems have been around us for generations. As a matter of fact, the "old" good linear regression model is, in a sense, also a prediction model. In its origin, the regression model is an explanatory model aimed at finding the relationship between an output variable, represented by the dependent variable, and a collection of domain-related input variables, often a small number of them, represented by the independent variables. The relationship between the variables could be either causal, often based on theory or practice, allowing one to affect the outcome of the output variable by changing the values of one or more of the input variables, or statistical that does not imply causal relationship but only expresses the strength of the relationships between the variables. A good explanatory model is a model with high fit between the output and the input variables that meet the assumptions underlying the regression models. If these assumptions are not met, one can apply transformations on the input and/or the output variables to render a model that meets the assumptions behind the regression model. Once a "good" model is found, one can also use it for predicting the value of the output variable for values of the independent variable which lie within the range of the input variables (interpolation). But one should be very cautious to use the model for predicting the values of the output variable for values of the independent variables which lie outside the range of the input variables (extrapolation).

Prediction problems are problems in which there exists a consequential relationship between the variables that enables one to predict the values of the output variable as a function of the input variables. The objective is to predict the values of the output events beyond the range of the input values. The idea is to build a model, based on past observations, from which we can discern the relationships between the output and the input variables, and then apply this model to predict the values of the output variable for new observations. Typical applications in marketing are targeting decisions for a product or a service where the objective of the model is to estimate the purchase probability of the customers. Then, instead of approaching all millions of customers in the database, most of them are not going to respond to the offer anyways, soliciting only the customers who are most likely to respond to the offer based on the predicted response probabilities. In targeting decisions, the model is often based on previous campaigns for the same or "similar" products. If a new product is involved for which no information exists yet on how customers will react to the offer, the model is based on a representative sample of customers drawn randomly from the database, who are approached with the offer to purchase the product/service. Then, based on the customers' response, building a response

model which is applied on the balance of the customers in the database in order to identify the customers who will take part in the marketing campaign. This method saves redundant marketing costs, therefore increasing profitability, because it avoids approaching the many customers who, based on the model, are not likely to respond to the marketing offer. This model is cross-sectional, being based on a sample of customers selected randomly across the entire database.

Another type is a temporal model where observations are drawn over time. A common example is a churn application which is based on a sample of customers in a certain time period, say the last quarter, some of whom have defected the company and some who have not, in order to build a churn model which is then applied to predict the churn probability of customers in the next quarter. This information enables the organization involved, e.g., a cellular company, to take actions ahead of time to avoid customers who were identified as potential churners by the model, certainly the profitable ones, to defect to a competitive company.

By and large, the PA process consists of two main processes: a model-building process that fits a model to the data, and a scoring process that uses the model to predict the response of new observations. There are many types of response models to entertain the variety of prediction problems which exist in practice. As mentioned above, in this article we focus on the PA process for binary (Yes/No) targeting decisions involving logistic regression.

3 How to Evaluate a PA Model?

Certainly, the logistic regression model is one of the leading models to predict a binary Yes/No (0/1) target variable. While most other prediction models, such as neural networks, decision trees, Bayesian methods, and others, are heuristic methods with no much theory behind them, the logistic regression model has a solid theoretical basis which allows one to interpret and explain the model parameters and results. The output of the logistic regression model, the so-called scores, are well-defined probabilities. In targeting buy-no-buy applications, for example, these scores represent the purchase probabilities of the customers. Furthermore, the sum of the predicted purchase probabilities across a group of customers in the training dataset (e.g., females) yields the predicted number of orders of that group, certainly an important output of any targeting application. The fact that the output scores of the logistic regression model are well-defined probabilities allows one to apply solid economic analysis to make better targeting decisions which are based on economic criteria. Furthermore, Hastie et al. (2009) have shown that one can estimate the parameters of a logistic regression model by solving a series of weighted linear regression models, which provides yet another strong theoretical foundation for the logistic regression model. All these criteria make the logistic regression model an ideal candidate as a prediction model. Thus, in the following we use the logistic regression model for actual predictions, but the theoretical discussion will be based on the linear regression model.

By and large, a regression model is basically an optimization model whose purpose is to find the best fit between the input and the output variables based on a certain criterion, e.g., the least squares criterion in linear regression, and the loglikelihood criterion in logistic regression. No wonder why such a model will yield accurate prediction results if applied on the same dataset that was used to build the model. But our objective is to estimate the output values for new observations which did not participate in the model-building process. So how does one evaluate the performance of a predictive model when applied on a new unseen dataset?

The common way to evaluate the quality of the prediction process is to partition the learning dataset (also referred to as the test dataset) into two independent and mutually exclusive and exhaustive datasets – a training dataset for building the model, and a holdout dataset, also referred to as the validation dataset, which is set aside, for validating the model. In this approach, one builds the model based on the training dataset and then applies the model results on the validation dataset to predict the values of the dependent variable. Since the actual values of the response variable in the validation dataset are known, one can compare the predicted results to the actual values to assess the quality of the prediction process. Several variations of this approach exist for validating a model, including cross-validation, k-fold validation, and others, in all of which there is a separation between the training and the validation datasets.

The quality of the prediction process is influenced by two main factors:

- The bias of the model originating from the specification errors
- Sampling error originating from the fact that the model parameters are estimated based on a sample

Hastie et al. (2009) have shown that for a squared loss function, one can decompose the expected prediction error, EPE, into three components:

$$\text{EPE} = E\left(Y - \hat{f}(X)\right)^2 = \underbrace{E(Y - f(X))^2}_{\text{Var(Y)}=\sigma^2} + \underbrace{[E(\hat{f}(X) - f(X)\Big]^2}_{\text{Bias}^2} + \underbrace{E\Big[\hat{f}(X) - E\left(\hat{f}(X)\right)\Big]^2}_{\text{Sampling Varaince}}$$

where:

Y is the target variable (the independent variable)
X is the vector of the independent variables
$Y = f(X) + \varepsilon$ is the regression model used to fit the data
ε is the error term satisfying $E(\varepsilon) = 0$ and $V(\varepsilon) = \sigma^2$
$\hat{f}(X)$ is the estimator of $f(X)$ based on the training dataset. One can estimate $\hat{f}(X)$ using a variety of supervised learning methods.

The first term in the EPE equation, $E(Y - f(X))^2 = \sigma^2$, expresses the variance of the target variable Y. This is a constant factor which we have no control on, even if we know the real $f(X)$ function.

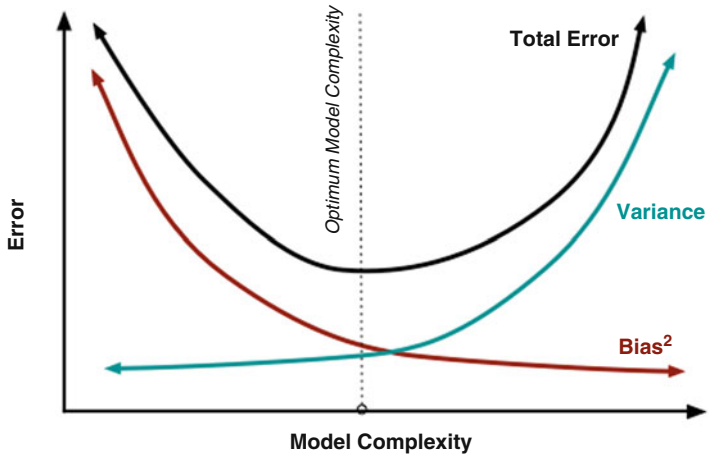


Fig. 1 The bias and sampling variance trade-off

The second term, $[E(\hat{f}(X) - f(X)\Big]^2$, expresses the squared bias of the model, or the specification error, originating from the difference between the true $f(X)$ function and its estimator $\hat{f}(X)$.

And finally, the third term, $E\Big[\hat{f}(X) - E\left(\hat{f}(X)\right)\Big]^2$, expresses the sampling variance, originating from the fact that the model is built based on a random sample.

The partition of the expected predicted errors to its component factors gives us a first glance on the difference between an explanatory and a prediction model. As mentioned above, we have no control on the first factor – the variance of Y. The second factor, the bias, is the factor that we try to minimize in building an explanatory model in order to attain the best fit of the model to the data. Prediction models, on the other hand, attempt to minimize the total error, i.e., the sum of the model bias and the sampling variance.

The main problem afflicting predictive analytics is the over-fitting phenomenon – a model which gives good prediction results when applied on the training dataset but poor predictions when applied on the validation dataset. This phenomenon becomes more serious when the number of the input variables gets larger, certainly a very critical issue in the world of big data. Geman et al. (1992) showed that there is a tradeoff between the bias of the model and the sampling variance as a function of the model complexity (Fig. 1).

The complexity level of a regression model is measured by means of the number of explanatory variables entering the model. On the one hand, the larger the number of variables entering the model, the better the model fits the data (the specification errors get smaller) and the higher the prediction accuracy on the training dataset. But on the other hand, the larger the number of variables entering the model, the larger are the chances that some insignificant variables will also enter the model, thus increasing the prediction error on the validation dataset.

The objective function in a prediction model is, therefore, not to build a model that yields maximum accuracy with "perfect" fit between the input and the output variables on the training dataset, but a model which minimizes the prediction errors, namely the sum of the bias and sampling variance errors (the total error). Or, putting this in different words, the objective of the PA process is to build a model which is generalized enough to be applied also on new observations that did not take part in the model-building process.

As we can see from Fig. 1, it is impossible to simultaneously decrease the specification errors and the sampling variance. A trade-off analysis between them is therefore required to yield a model which is accurate enough, yet minimizes the prediction errors for new observations. We demonstrate this process below for the case of linear regression.

4 How to Select the Explanatory Variables for the Model?

The most complicated problem in building large-scale multidimensional regression model is selecting the most influential explanatory variables for the model from among a much larger set of potential predictors. Data mining people refer to this problem as the feature selection problem, and statisticians as the specification problem (Miller 2002). PA models in the world of big data are characterized by a very large number of potential predictors, often hundreds, even thousands, which render the feature selection problem a most complex combinatorial problem.

When the number of explanatory variables is relatively small, say less than 10, one can select the most influential explanatory variables using a trial & error process. But when the number of potential explanatory variables is larger, certainly when dealing with big data problems, there i's no escape but to use automatic or heuristic methods to find the most influential predictors to introduce to the model. Indeed, a variety of methods have been developed in recent years to deal with the feature selection problem for a regression model, including "naïve" methods which filter out the explanatory variables using heuristic methods (e.g., the top 40 variables with the highest correlation with the dependent variable), StepWise regression methods (to be detailed later), dimensionality reduction methods such as PCA (principal component analysis) which represent the explanatory variables by means of a few "super" variables thus reducing the dimensions of the PA problem, regularization methods that "penalize" the objective function in order to decrease the model complexity (such as ridge regression or the LASSO approach), and many others. All these methods, and more, are widely discussed in the book of Hastie et al. (2009).

Undoubtedly, the StepWise Regression (SWR) method (Efoymson 1960) is by far the most common approach for automating the feature selection process in regression models. It is easy to use and is supplied with most commercial statistical software (SAS, SPSS, R, . . .). Several variations of SWR exist: the forward selection algorithm which starts with zero variables in the model and

then adds the most significant variables to the model at each iteration until the termination criteria are achieved. The backward selection algorithm which starts with all variables in the model and then removes the least significant predictors at each iteration until the termination criteria are achieved. And the forward/backward SWR model which combines both of these methods by iteratively adding and removing predictors until convergence.

All SWR models use *per-comparison* methods, testing one predictor at a time (Tukey 1977). The main disadvantage of this approach is that it tends to introduce too many predictors into the model. Tukey (1977) refers to this as the selection effect. Diaconis (1985) claims that "if enough statistics are compared, some of them will be sure to show structure". In other words, even if the set of potential predictors consists of random variables which have got nothing to do with the response, if this set is large enough, some of them will erroneously be found significant just by pure chance. It is well known that using a *significance level* $\alpha = 5\%$, one out of 20 insignificant predictors will be introduced, on average, erroneously into the model (type I error). However, when multiple tests are involved, the probability of making a type I error for any single predictor also grows. Using simple probability rules, it can be shown that in a *per-comparison* method involving k independent tests, the probability of making at least one type I error (i.e., declaring an insignificant predictor as significant) is $1 - (1 - \alpha)^k$, which is much larger than the required level of significance. For example, if $\alpha = 5\%$ and $k = 10$, the type I error probability is close to 40% (!), introducing many insignificant predictors to the model. One can reduce type I error by reducing the level of significance. But this will cause fewer significant predictors to enter the model and may increase the number of insignificant predictors in the model thus increasing the type II error.

Unfortunately, there is no way to reduce both type I and type II errors. On the contrary, decreasing the type I error usually increases the type II error, and vice versa. Hence a trade-off analysis between these two types of errors is required to find the best subset of predictors to introduce to the PA model. Below we discuss two alternative approaches that have been proposed in the literature to adjust the level of significance to reduce the errors involved – *Bonferroni* and *False Discovery Rate* (*FDR*).

Bonferroni Adjustment

The *Bonferroni* adjustment belongs to the class of the Familywise Error Rate (FWER) methods, which is the probability of making a single Type-I error in a series of k tests (Hochberg and Tamhane 1987). This method controls the probability of erroneously rejecting even one of the true null hypotheses or accepting a model with even one erroneous predictor. A *Bonferroni* procedure with a significance level α is equivalent to a *per-comparison* procedure with significance level $\alpha^* = \alpha / k$, where k is the number of simultaneous hypothesis comparisons. For example, in a feature selection problem for linear regression for which there are still 10 more predictors to consider for introducing to the model, using FWER with *p-value* smaller than 5% is equivalent to *per-comparison* error with *p-value* level smaller than 0.5% (= 5%/10). Reducing the level of significance is bound to reduce the Type-I error, because it

will only introduce predictors which are highly significant. But when the number of potential predictors is large, the *Bonferroni* modification results in significance levels which are quite restrictive, thus eliminating important predictors from the model and letting more insignificant predictors in, thus increasing Type-II error.

False Discovery Rate (FDR)

So, *per-comparison* methods are too "loose" allowing too many predictors into a model, increasing Type-I error. FWER methods, like *Bonferroni*, are too "tight" excluding important predictors from the model, increasing Type-II error. The *FDR* approach offers a compromise by controlling the type I error rate based on the number of predictors already in the model (Benjamini and Hochberg 1995). We note that the terminology behind the FDR method is taken from the world of scientific discoveries where the null hypothesis is that there is no scientific discovery versus the alternative hypothesis that there is. Thus, the term "false discovery" corresponds to the case where we reject the null hypothesis that there is no discovery (with a certain probability) and accept the alternative hypothesis that there is a discovery even though it is not true.

Unlike the traditional SWR approach, the FDR approach controls the false discovery rate rather than the chance of any false discovery. Thus, if the number of significant predictors is relatively small, FDR uses tight significance levels in order to introduce most of them to the model. However, if the number of significant predictors is relatively large, FDR relaxes the significance level in order to "capture" as many of them even if it means introducing a few insignificant predictors to the model. The idea behind the FDR approach is that if the number of significant predictors is small, even a small number of insignificant predictors entering the model may distort the model. But if the number of significant predictors is relatively large, no much harm will be caused if some insignificant predictors will also enter the model. This is definitely better than reducing the level of significance and introducing only a fewer number of significant variables to the model.

We note that the approach of incorporating FDR within the SWR process is slightly different from the original FDR method of Benjamini and Hochberg (1995). The original approach tests the significance of all the predictors simultaneously in a single multi-comparison process using the FDR thresholds to select the influential predictors. This approach may not be suitable for multi-variate regression models in the business world which are characterized by high correlation between predictors. However, by combining SWR, which is better capable of coping with the multicollinearity issue, with the FDR approach, which balances between type I and type II errors, we can enjoy the best of all worlds – dealing explicitly with the multicollinearity issue while at the same time controlling the error probabilities. The clue is to appropriately change the level of significance at each of the SWR iterations. This may be accomplished by means of a search procedure by running the SWR method, at each step of the process, for several ranges of (P_e) (the p-value for entering a predictor to the model) and (P_d) (the p-value for deleting a predictor from the model), and then modifying the corresponding FDR value for each based on the number of predictors that already entered the model.

5 Explanatory Model Versus Prediction Model

The above discussion raises the question whether there is a difference between an explanatory model and a prediction model, and whether one may use an explanatory model to predict the dependent variable for new observations which lie outside the range of the explanatory variables that entered the model? Indeed, by the statistical literature, there is a fundamental difference between a model aimed at explaining a given phenomenon and a model aimed at predicting a given phenomenon for new observations. These differences also affect the way of building a prediction model versus an explanatory model. According to Shmueli and Koppius (2011) and Shmueli (2010), three main factors are responsible for the differences between an explanatory and a prediction model. The first is that explanatory models are usually based on causal relationships (often based on theory) or statistical relationships between variables, whereas prediction models are based on associative relationships between measurable variables. The second difference relates to the objective function of each model. While an explanatory model seeks the model that maximizes the fit for the data that took part in building the model (i.e., reduces the bias), a prediction model seeks the model that minimizes the total error (the bias plus the sampling variance). By Fig. 1 there is a trade-off between the model accuracy level and the prediction accuracy level which implies that one may sometimes need to give up model accuracy in order to attain a better prediction accuracy. In other words, a good explanatory model is not necessarily a good prediction model, and vice versa. And, finally, the third difference between an explanatory and a prediction model is that a prediction model is a proactive model for predicting unseen events, while an explanatory model is a retrospective model aimed at explaining a given phenomenon based on past and present data. Prediction models require that all the variables that entered the final prediction model must also be available for the new observations participating in the prediction process, while there is no such constraint for an explanatory model. Table 1 summarizes the differences between a model for explaining a given phenomenon and a model for predicting new observations (according to Shmueli 2010):

6 So How Does One Build a Good Prediction Model?

Since there is no way to determine upfront the explanatory variables that yield good model that is also general enough to apply to new observations, the process of building a prediction model is based on a trial & error procedure involving various measures for evaluating model and prediction accuracy. The idea is to control the complexity level of the model and find the "optimal" number of variables that will yield an accurate model yet provide reasonable predictions when applied on unseen data. As shown in Fig. 1, the priority is to build a parsimonious model that might be less accurate but yield good prediction results.

Table 1 Explanatory versus prediction models

Item	Explanatory model	Prediction model
Analysis goal	Find the "best" relationships between the input and the output variables	Predicting events for new observations
Variables	Based on theory and domain knowledge, usually a small number of variables and relatively small samples	Observed measurable variables, usually many of them and large samples
Objective function	Maximize fit, high accuracy, minimize bias	High predictive power, minimize total sample error (bias and variance)
Model building constraints	Easy to interpret, supports statistical tests of hypotheses, adhere to theoretical model	Use explanatory variables that are also available for the new observations
Model evaluation	Explanatory power is measured by the degree the model fits the data, such as significance tests, R^2, variable coefficients	Prediction power is measured by means of the accuracy of predictions for out-of-sample data
Time frame	Retrospective	Prospective

In this section, we discuss a few more complementary methods that can be incorporated within the StepWise regression procedure in order to control the type I and type II error probabilities and improve prediction accuracy.

- R^2 – R-Squared
  R-Square (R^2) , also known as the coefficient of determination, is undoubtedly the most common measure to evaluate the goodness of fit of a regression model. R^2 expresses the proportion of the variance explained by the regression model. It is a measure in the range 0–1. The closer R^2 is to 1, the better the fit of the model to the data, and vice versa, the closer R^2 is to 0, the worse is the fit. The main problem with the R^2 criterion is that it gets better, or at least doesn't change, as we add more predictors to the model, even noisy and irrelevant ones, thus increasing model complexity and bringing us to the region in Fig. 1 beyond the minimal point of the total error where the prediction error increases. While R^2 is a good measure for attaining high model accuracy, it is therefore less appropriate for building prediction models.
- $\overline{R^2}$ - adjusted R^2
  An alternative measure is a modified version of R^2 that also accounts for the number of variables that were already introduced to the model. The adjusted R^2, often denoted by $\overline{R^2}$, is perhaps the most popular of these modified methods. The adjusted R^2 adds a penalty function that decreases as more predictors are added to the model:

$$\overline{R^2} = 1 - \left(1 - R_k^2\right) \frac{n-1}{n-k-1}$$

where:

- n is the number of potential variables
- k is the number of variables already in the model
- $\overline{R_k^2}$ is the adjusted R^2 with k variables in the model

  When $k = 0$, the penalty is 1 and $R^2 = \overline{R^2}$. When $k \geq 1$, $R^2 \geq \overline{R^2}$. As we increase the number of predictors, the penalty function $(n - 1)/(n - k - 1)$ decreases and only significant changes to R^2 will increase $\overline{R^2}$ (Miller 2002). Indeed, using the adjusted R^2 criterion usually reduces the number of variables entering the regression model, thus making it a preferable criterion for selecting variables to a regression model.
- Information-based methods

  These methods measure the information loss between the true (but unknown) model that generates the data (e.g., a linear model), and the model estimated based on the sample data (e.g., the regression model). The less information is lost, the better the quality of the estimated model. The leading measures in this category are based on the AIC – Akaike Information Criteria (Akaike 1974):

$$\text{AIC} = m * k - L_k$$

where:

- $m = 1,2,\ldots$ a constant integer
- L_k is the Log-Likelihood function of a model with k parameters

The objective is to minimize the value of the AIC. The first expression $m*k$ is a type of penalty that depends on the number of parameters (variables) in the model. The default value of the constant m in the AIC expression is 2. But one can pick a constant m which is larger than 2. The larger the value of m, the larger is the penalty and the smaller the number of variables entering the model.

A variation of AIC is the BIC – Bayesian Information Criterion (Schwartz 1978):

$$\text{BIC} = m * \text{Log}(k) - \text{Loglikelihood}$$

The BIC imposes even a higher penalty on the model complexity and reduces the number of variables entering the model even further.

Other information-based criteria also exist but they are not as common as the above criteria.

- Significance-based Criteria

  In addition to experimenting with different goodness-of-fit criteria, one can affect the complexity of the prediction model by means of the level of

significance. We've already mentioned the Bonferroni and the FDR criteria. Another way is to deviate from the common level of significance of 5% to build a model (say, 10%) that might yield a less accurate model, but will render a model that yields better predictions for new observations. Another alternative is to use different significance levels depending on the variables' type, e.g., one significance level for continuous variables and another one for discrete and categorical variables. Another option is to vary the level of significance based on domain knowledge or even intuitive considerations.

- Selecting groups of predictors instead of individual predictors
  Transformations of explanatory variables are a popular means to decrease model bias. A common transformation is binning which represents continuous variables by means of a series of 0/1 categorical variables. For example, binning of a continuous variable based on quartiles will yield 4 categorical predictors, one for each quartile, each is set to 1 if the value of the continuous variable for the observation falls in the corresponding quartile, and zero otherwise. Another example is the marital status with four values – single, married, divorced and widowed, which is represented by means of four categorical predictors each of which is set either to 0 or 1 depending upon the observation's marital status.
  These transformations usually increase the number of potential predictors in the model-building process. Introducing/removing individual predictors from the model in the SWR process is likely to increase the model complexity which may cause over-fitting and decrease prediction quality. To overcome this issue, it is recommended to select the explanatory variables to the model on a group basis, rather than on an individual basis. In the marital status example, for instance, this means introducing all 4 categorical predictors to the model if marital status significantly improves model quality, even if one or more of the individual categorical predictors is not significant. Similarly, remove all 4 categorical predictors from the model if the marital status does not improve model quality, even if one or more of the individual categorical predictors is significant. Introducing/removing variables on a group basis may severely complicate the SWR process for building a regression model but it is certainly worth the effort as it yields a more stable model.

7 Performance Measures for Assessing Prediction Accuracy

The above represents only a partial list of methods to improve prediction accuracy. Since it is not known in advance which method works best, there is no escape but to experiment with alternative model configurations and various parameters and goodness-of-fit criteria, in order to converge to the "best" model. We emphasize again that what we seek here is not only a model that renders good predictions for unseen and new data, but also a model which is accurate enough and fits the data well. For this, we need performance measures that would allow one to assess the model fit as compared to its prediction accuracy.

To recall, the model's accuracy level is represented by means of the variance of the predicted values of the dependent variable for the training dataset, and the prediction accuracy level by means of the corresponding variance on the validation dataset. Hence, one way to assess the quality of a model is to compare the value of R^2 between the training and the validation datasets. A "good" model is one for which the R^2 values for the two datasets are "close" enough. Instead of R^2, one can use other goodness-of-fit criteria such as $\overline{R^2}$, AIC, BIC, and others.

But all these criteria measure the overall quality of the model, namely how well the model fits the data, which may not suffice for making actual decisions. In targeting applications, for example, we are interested in assessing the gains achieved by the model, or how well the model discriminates between targets and non-targets. The objective is to approach only the "good" customers, namely the customers who are most likely to respond to a solicitation to buy a product/service. This way, one can avoid approaching most of the customers who are less likely to respond to the offer and thus save a lot of marketing costs. And in churning problems the objective is to differentiate between churners and non-churners and approach only customers who are more likely to churn, certainly the profitable ones, with an offer that may convince them to stay with the company. For this we need more detailed performance measures which are based on the distribution of the targeting results, known as the gains table, or its graphical representation – the gains chart. We describe these measures in more detail below. While these measures are taken from the marketing domain, they can also be applied to other domains as well. To be consistent with the discussion above, these measures also correspond to targeting decisions based on binary logistic regression models.

Gains Table

The gains table exhibits the model performance results in a tabular form. The gains table is created by sorting out the customers in descending order of the score they receive by the model and grouping the observations at a decile or any other percentile level. In targeting decisions involving logistic regression, the score of each observation represents the probability that the customer will respond to the purchase offer. Ordering the customers in decreasing order of their purchase probabilities will therefore place the "good" customers with the high probability of purchase at the top of the list and the less likely to respond people at the bottom of the list. Table 2 presents a sample gains table for a targeting application.

The model results In Table 2 are presented at the decile level in decreasing order of the response probabilities estimated by the logistic regression model. The left-most column (RESPONSE PROB. %) expresses the lower bound of the predicted response probabilities, at the corresponding decile. For example, the lower bound of the purchase probability in the top decile is 22.08%, meaning that the response probabilities of the customers in the top decile is larger than 22.08%. The response probabilities of customers in the second decile is between 12.42% and 22.08%, and so on. Table 2 also details, at the decile level, the following data:

Table 2 Gains Table, Logistic Regression Model, Validation Dataset

RESPONSE PROB %.	CUST.	% CUST.	RESP	% RESP	ACTUAL RESPONSE RATE %	% RESP. / % CUST.	PREDICTED RESPONSES	PREDICTED RESPONSE RATE %
22.08	1647	10.0	660	43.2	40.07	4.3	677	41.07
12.42	1647	10.0	270	17.7	16.39	1.8	270	16.39
8.19	1647	10.0	174	11.4	10.56	1.1	168	10.19
5.90	1646	10.0	136	8.9	8.26	0.9	114	6.95
4.32	1647	10.0	93	6.1	5.65	0.6	84	5.09
3.25	1647	10.0	62	4.1	3.76	0.4	62	3.79
2.34	1646	10.0	44	2.9	2.67	0.3	45	2.71
1.61	1647	10.0	46	3.0	2.79	0.3	32	1.94
0.89	1647	10.0	30	2.0	1.82	0.2	21	1.27
0.03	1646	10.0	14	0.9	0.85	0.1	8	0.51

CUST. is the number of customers
% CUST. is the percentage of customers
RESP. is the number of actual responders
% RESP. is the percentage of actual responders
ACTUAL RESPONSE RATE % is the actual response rate
% RESP./%CUST. is the ratio between the actual response rate and the percentage of customers
PREDICTED RESPONSES is the number of responders predicted by the model
PREDICTED RESPONSE RATE % is the predicted response rate

Using Table 2 one can easily assess the prediction quality of the model by means of several measures. For instance, the number of actual responders is a monotonically decreasing function when one traverses across the deciles, indicating that the model was successfully able to place the good customers, with the high purchase probabilities, at the upper deciles and the less-likely-to-respond customers at the lower deciles. Indeed, the number of actual responders in the top decile is 660 which constitutes 43.2% of the total number of responders with a response rate of 40.07%, versus only 14 responders in the bottom decile, which constitutes only 0.9% of the responders with a response rate of 0.85%. A similar pattern also exists for the predicted number of responders. For example, the model predicts 677 responders at the top decile which makes up 41.07% of the total number of predicted responders, versus 8 predicted responders in the bottom decile which makes up only 0.51% of the predicted responders. Additional measure for the quality of the model is how well the model can predict the responders versus the actual number of responders. Indeed, comparing the actual number of responders to the predicted number of responders reveals close proximity at the decile level. For example, the actual number of responders in the top decile is 660 and the number of predicted responders is 677. In the second decile, the number of actual responders is 270, the same as the number of predicted responders, and so on.

Gains Chart

The Gains chart exhibits the model results in a graphical form, displaying the added gains (e.g., profitability or response) in using a predictive model versus a null model that assumes that all customers are alike. Figure 2 exhibits the gains chart corresponding to the data in Table 2. The horizontal axis represents the cumulative percentage (or proportion) of the population (prospects), the vertical axis the cumulative percentage (or proportion) of the actual responders, where the customers are ordered in descending order of the predicted values of the dependent variable (the scores), i.e., $\hat{y}_i = \hat{y}_{i+1}$, where $\hat{y}_i$ is the score estimated by the model for customer i.

The diagonal 45° line represents the null model, namely the results that we expect if no targeting model is used. Thus, in a marketing campaign if we approach, say, 30% of the customers at random, we expect to capture about 30% of the responders. But if we approach the best 30% of the customers as identified by the model (namely, the customers at the top three deciles), we expect to capture almost 75%

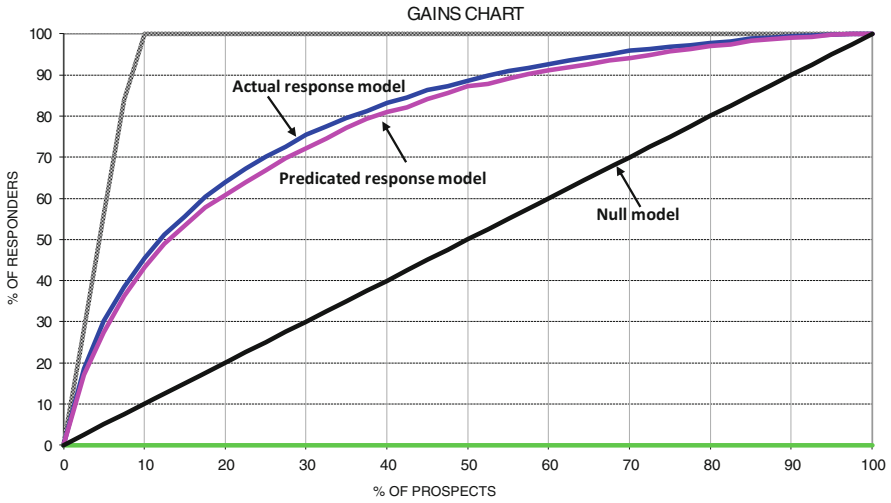


Fig. 2 Gains chart

of the responders, quite a hefty lift. Similarly, if we approach 50% of the customers randomly, we expect to capture about 50% of the responders, but if we approach the best 50% of the customers as identified by the model, we expect to capture almost 85% of the responders. And so on. The curve above the null model exhibits the predicted responders, the curve above it is the model curve exhibiting the actual responders. As can be seen, the two curves are pretty close to one another indicating that the model is not only doing a good job in identifying the better customers, but it also doing a good job at predicting the number of responders. The larger the gap between the model curve and the null model, the better the prediction model. The curve at the top of the gains chart represents the "perfect" model, namely the model that captures all the responders, all of whom are included at the top deciles. Of course, one can never find that perfect model, but it serves as a reference curve for assessing the quality of the model results, the closer the model curve to the perfect model curve, the better the model.

Two more metrics, based on the gains chart, are typically used to assess how the model predictions differ from the null model:

- Maximum Lift (*M-L*), more commonly known as the Kolmogorov-Smirnov (K-S) statistics (Lambert 1993) which is the maximum distance between two distributions (e.g., the model curve and the null model). The K-S statistics has a distribution known as the *D* distribution (DeGroot 1993) which allows one to test the hypothesis that an empirical distribution function is drawn from a given reference distribution (the one sample test) or that two empirical distributions are drawn from the same distribution (the two-sample case). In our case, the K-S test can be used to test the hypothesis that the model curve is significantly different than the null model, which indicates that the model discriminates well

between targets and non-targets. Another application is to test the hypothesis that the actual responders curve and the predicted responders curve come from the same distribution, which provides yet another indication on the quality of the model because it implies that the model is also doing a good job in predicting the number of responses at each decile level.

- The Gini coefficient (Lambert 1993) expresses the area enclosed between the model curve and the null model, often divided by the area below the null model. In most applications, a large Gini coefficient indicates that the model curve is different from the null model.

An alternative way to assess the quality of a prediction model is by means of the ROC (Receiving Operating Characteristics) curve and the AUC – the Area Under the ROC. The ROC is equivalent to the Gains chart and the AUC to the Gini coefficient.

The K-S statistics, the Gini coefficient, and the AUC are all scalar values and are often used to also compare the performance of two alternative models applied on the same datasets.

Finally, we note that the validation process discussed above applies primarily to a cross-sectional situation where both the training and the validation datasets are drawn from the same universe and at the same time frame. A typical application is in database marketing where a sample of customers are selected from the universe (the test data) and then partitioned into a training and validation datasets for building and validating the model. This approach may be less suitable for dynamic applications, such as streaming data and IOT, where the dynamics of the system, changing business environment, changes in the input data, aging of models over time, and others, may affect the model performance to an extent that it becomes inappropriate, even obsolete, for scoring new observations. In these cases, one needs to closely monitor the model performance over time and issue alerts whenever the model performance degrades to a degree that requires modifying the model. This topic falls outside the scope of our article but is handled extensively in the data streaming mining environment. A good reference is the book by Bifet et al. (2018).

8 How to Detect Over-Fitting?

Indeed, all performance measures above, which correspond to the validation dataset, indicate a good model. But what is the guarantee that the model does not "suffer" from the over-fitting phenomenon which often afflicts multi-dimensional prediction models? For this, we need to compare the model results between the training and the validation datasets. In Table 3 we present several performance measures to assess over-fitting at the decile level. For easy comparison, the performance measures for the two datasets are displayed side-by-side, where TRN denotes the training dataset and VAL the validation dataset.

Looking at the actual RR (%) columns in Table 3, the actual response rate for the training dataset at the top decile is 40.61 which is very close to the actual response

Table 3 Performance measures for the training and validation datasets

DECILE	TRN % RESP.	VAL % RESP.	TRN % RESP. / % CUST.	VAL % RESP. / % CUST.	TRN ACTUAL RR %	VAL ACTUAL RR %	TRN PREDICTED RR %	VAL PREDICTED RR %
1	45.5	43.2	4.5	4.3	40.61	40.07	40.71	41.07
2	18.4	17.7	1.8	1.8	16.43	16.39	16.09	16.39
3	11.6	11.4	1.2	1.1	10.35	10.56	10.10	10.19
4	7.7	8.9	0.8	0.9	6.83	8.26	6.96	6.95
5	5.3	6.1	0.5	0.6	4.77	5.65	5.10	5.09
6	4.2	4.1	0.4	0.4	3.76	3.76	3.80	3.79
7	3.1	2.9	0.3	0.3	2.80	2.67	2.74	2.71
8	1.9	3.0	0.2	0.3	1.73	2.79	1.97	1.94
9	1.6	2.0	0.2	0.2	1.43	1.82	1.28	1.27
10	0.6	0.9	0.1	0.1	0.54	0.85	0.51	0.51

rate for the validation dataset – 40.07%, the corresponding values for the second decile are 16.43% and 16.39%, for the sixth decile the response rate is 3.76% in both files. And so on, the actual response rates at the decile level for the training and validation datasets are similar. Likewise, the predicted response rates for the training and validation datasets are also close enough. A similar phenomenon appears for the other measures. This is a clear indication of no over-fitting. As to the proximity level to rule out over-fitting, a rule of thumb is that a deviation in the range of 5–10% is quite reasonable. But one can use more accurate statistical tests to assess over-fitting, for example test the hypothesis that the average response rates, across all deciles, between the training and the validation dataset are equal, or apply the two-sample K-S test to test the hypothesis that the corresponding distributions of the results for the training and validation datasets are drawn from the same distribution.

9 Economic Considerations and Decision-Making

Finally, after checking the model quality and making sure we obtain a stable prediction model which is accurate enough with no over-fitting, the next stage is to deploy the model results to predict events for new observations. We use below a direct mail example to demonstrate the decision-making principles. But these principles may also apply to other domains.

As mentioned above, the PA process consists of two principal components – model building and scoring. So far, we concentrated only on the first component having to do with building a good-quality prediction model. The next stage in the PA process is to apply the prediction formula coming out from the first stage (e.g., a regression equation) on the database which did not participate in the model-building process, or on new observations, to calculate a "score" for each observation. We recall that in the case of a binary logistic regression model with a 0/1 dependent variable, the score of each observation, in a targeting application, expresses the predicted purchase probability of the customer, the higher the predicted probability the higher the likelihood that the customer will respond to a solicitation to purchase the product/service offered to him/her, and vice versa. Sorting the customers in descending order of the predicted purchase probabilities will therefore place the "good" customers at the top of the list and the "bad" customers at the bottom of the list.

The main decision problem is how far down the list one can go to include customers in the marketing campaign and still be profitable? Or, in other words, what is the cutoff response rate (CRR) which separates out the targets from the non-targets? In the lack of economic considerations, a common way to select the CRR is either based on domain knowledge or to approach the customers in the top 4–5 deciles of the predicted purchase probabilities.

Clearly, a better way to select customers for the marketing campaign is based on economic considerations. The idea is to approach only the customers for whom

the expected profit by purchasing the product/service is greater than or equal to the contact cost of approaching the customer. We denote:

c – the contact cost (which consists mainly of the mailing cost and the brochure cost)
g – the net profit per unit of the product (often the unit price minus the manufacturing cost, the shipment cost of the product to the customer's house, discounts, packaging costs, . . .)
$\hat{p}_i$ – the predicted purchase probability of the ith customer

The transaction is profitable if the expected profit from the customer is greater than or equal to the cost of approaching the customer (the investment cost), namely:

$$\hat{p}_i * g \geq c \Rightarrow \hat{p}_i \geq c/g$$

And from here it is easy to obtain the CRR (in percentage terms):

$$\text{CRR} = 100 * c/g$$

For example, if the cost of approaching the customer is \$0.95 and the profit per unit of product is \$120, the CRR is 0.79%, implying that it is worth approaching only the customers whose predicted purchase probabilities, coming out from the PA model, is equal or greater than 0.79%.

Figure 3 exhibits the economic calculations in a graphical form for the gains table data in Table 2. But rather than presenting the results at the decile level, the economic calculations were conducted at percentile levels of 2.5%. The X-axis presents the cumulative percentage of customers. The Y-axis presents the expected profit (in \$). For each percentile level, the expected profit is obtained by the sum of the products $\hat{p}_i * g$ over all the customers in the percentile, minus the contact cost c times the number of customers in the percentile. The calculation in Fig. 3 is presented both for the actual profits and the predicted profits.

As seen from Fig. 3, the expected profit initially increases as a function of the percentile level, until it reaches a maximal point, from which it declines until it reaches the zero point and turns to be negative. Contacting the top 4 deciles yields positive expected profits. Contacting customers beyond the 4 top deciles is not worthwhile as it yields negative profits. Maximal profit is obtained for the top 15% of the customers. Note how close are the actual profits to the expected predicted profits, certainly another clear indication of the quality of the model.

In the above discussion, we assumed that the economic parameters (c, g) are the same for all customers. More realistic applications are those where each customer may have its own economic parameters (e.g., the profit per unit of sold product may vary by gender, the shipping cost of the product to the customer may vary by location, and so on). One can expand the above discussion also for the case of "personal" economic parameters. But at any case, the calculations are still based on the predicted purchase probabilities coming out from the PA model.

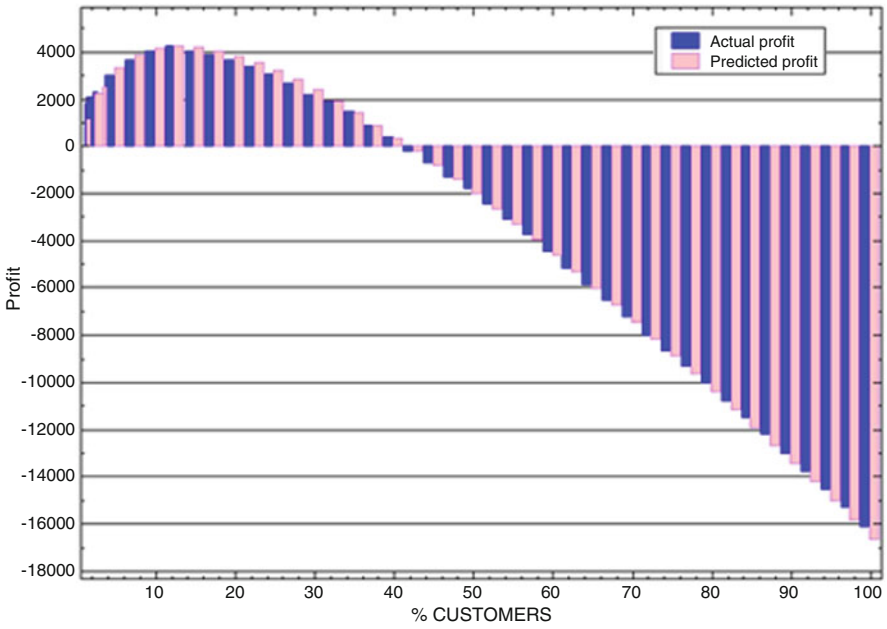


Fig. 3 Actual and predicted expected profits

10 Using Decision Trees for Feature Selection and Prediction

Regression models are parametric prediction models. Several non-parametric models have also been proposed for feature selection and prediction, one of the most popular of which is decision trees, which we briefly discuss in this section. More comprehensive description of decision trees can be found in the literature, for example, Murthy (1998).

A decision tree is a non-parametric heuristic method to partition an audience into "homogenous" classes (segments, groups) based on a discrete target variable, which may assume several predefined categories, usually a small number of them. The most popular case is the two-way classification where the target variable assumes only two possible categories, either Yes or No (e.g., targets and non-targets). The objective of the classification process is to partition the audience into mutually exclusive classes which are as pure as possible with respect to the target variable. A class is said to be totally pure if all observations in the class have the same target value. In the multi-attribute application, which is usually the situation in the world of big data, it is very rare, if not impossible, to find an attribute that splits an audience perfectly. Even if one subgroup may happen to be pure, the others are not. Thus, one cannot get by with only one split and needs to build a whole tree for this.

Starting from a "root" node (the whole population), tree classifiers employ a systematic approach to grow a tree into "branches" and "leaves". The process is

iterative. In each stage of the process the algorithm looks for the most informative attribute to split the current "father" node into several "children" nodes which are as pure as possible. Then, using a set of predefined termination rules, some nodes are declared as "undetermined" and become the father nodes in the next stages of the tree development process, some others are declared as "terminal" nodes. The process proceeds in this way until no more node is left in the tree which is worth splitting any further. The terminal nodes, or "leaves", define the resulting classes. The process results in a tree-like diagram. If each node in a tree is split into 2 children nodes only, one of which is a terminal node, the tree is said to be "hierarchical".

Several criteria have been proposed in the literature to split a node. The objective is to maximize the improvement in the node value by splitting it into two or more splits. In the case of targeting applications with two categories – targets (or buyers) and non-targets (non-buyers), the value of a node t is a function of the response rate (RR) of the node, RR(t), (i.e., the proportion of targets in the node). The "ideal" split is the one which partitions a father node into two children nodes, one which contains only buyers, i.e., RR(t) = 1, and the other only nonbuyers, RR(t) = 0. Clearly, the "worst" split is one which results in the two children having more-or-less the same proportion of buyers and nonbuyers, i.e., RR(t)~1/2.

These requirements define a family of node value functions, which we denote by Q(RR), which satisfy the following conditions:

- Max $Q(\mathrm{RR}) = Q(0) = Q(1)$
- Min $Q(\mathrm{RR}) = Q(1/2)$
- Q(RR) is symmetric, i.e. $Q(\mathrm{RR}) = Q(1 - \mathrm{RR})$
- Q(RR) is a concave function of RR.

The first two conditions stem from our definition of the "best" and the "worst" splits; the concavity and symmetry condition also follows from these conditions and the condition for the worst split.

Clearly, there are many functions that satisfy these requirements, including:

The piecewise linear function: $Q\ (\mathrm{RR}) = \begin{cases} \mathrm{RR} & \mathrm{RR} \leq 1/2 \\ 1 - \mathrm{RR} & \mathrm{RR} > 1/2 \end{cases}$

The quadratic function: $Q(\mathrm{RR}) = (\mathrm{RR} - 1/2)^2$

The entropy function: $Q(\mathrm{RR}) = -\ [\mathrm{RR}\ \log_2(\mathrm{RR}) + (1 - \mathrm{RR})\ \log_2(1 - \mathrm{RR})]$

And others.

We note that the above definition of the node value functions applies to applications where the number of buyers is more-or-less equal to the number of non-buyers. Clearly this condition does not apply in many marketing campaigns (e.g., in database marketing) where the number of buyers is much smaller than the number of nonbuyers. In these cases, the reference point of ½ may not be appropriate for defining the node value functions. A more suitable criterion to define the worst split in these cases is TRR, where TRR is the overall response rate of the test audience. This still results in the node value function being concave but non-symmetric.

The entropy measure from information theory (Shannon 1948) has gained a lot of popularity in recent years for building decision trees, and constitutes the basis for several common tree classifiers, including CART –- Classification and Regression Trees (Breiman et al. 1984), ID3 (Quinlan 1986), C4.5 (Quinlan 1993), and others.

Depending upon the attribute involved, the resulting value by partitioning a father node is obtained as the average values of the descendant nodes, weighted by the proportion of customers in the node, i.e.:

$$\frac{N_1}{N} Q\,(\mathrm{RR}_1) + \frac{N_2}{N} Q\,(\mathrm{RR}_2) + \frac{N_3}{N} Q\,(\mathrm{RR}_3) + \ldots$$

where:

N	is the number of customers in the father node
$N_1, N_2, N_3, \ldots$	is the number of the customers in the descendant nodes
$\mathrm{RR}_1, \mathrm{RR}_2, \mathrm{RR}_3, \ldots$	are the response rates of the descendant nodes.
$Q(\mathrm{RR}_1), Q(\mathrm{RR}_2), Q(\mathrm{RR}_3), \ldots$	is the corresponding node value functions.

Then the improvement in the node value resulting by the split, referred to as the Information Gain (IG), is given by the difference:

$$\mathrm{IG} = Q\,(\mathrm{RR}) - \left[\frac{N_1}{N} * Q\,(\mathrm{RR}_1) + \frac{N_2}{N} * Q\,(\mathrm{RR}_2) + \frac{N_3}{N} * Q\,(\mathrm{RR}_3) + \ldots\right]$$

To find the best split for each father node, one needs to go over all possible attributes in the dataset to split a father node by, calculate the resulting IG for each and then select the split that maximizes the IG value. Since the entropy function Q(RR) is a concave function, any split will result in a positive IG, however small. Thus, one needs to apply termination rules to stop the splitting process, such as restrict the number of observations and/or the number of targets in a node, set a minimum level on the IG to split a node, etc., or otherwise the algorithm will keep partitioning the tree until each node contains exactly one observation.

A simple decision tree is described in Fig. 4, where the decision is either to grant a customer a mortgage (OK) or not (Delay). Note that the Age attribute with two values (Age$\geqslant$50 and Age < 50) is split into two branches whereas Residence-type with three values (Own, Rent, and Other) is split into 3 branches. The tree diagram also exhibits a sample observation (ID # 87594) to predict whether to grant the customer a mortgage based on his/her attributes.

The advantage of a decision tree is that it can be used in two capacities, either as a feature selection process, or as a stand-alone prediction model. The main use of decision trees in practice has been as stand-alone prediction models in many domains, primarily because decision trees are easily interpretable and explainable. Trees are also displayed visually, which make it easy to track the path of any observations from the root node to its terminal node. For example, in the decision tree above, to determine whether to approve or deny the mortgage from customers

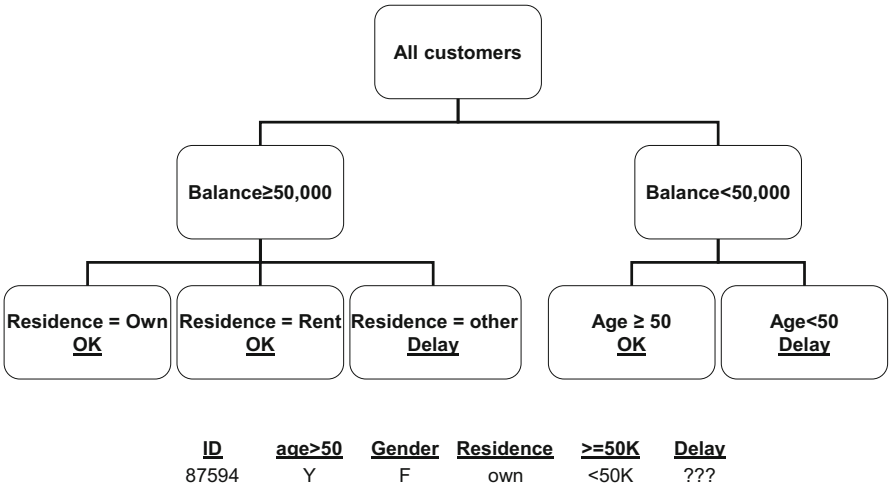


Fig. 4 A sample decision tree

based on their attributes. Decision trees can also be used to estimate probabilities. For example, in a buy/no-buy application, the purchase probability of customers is estimated by the response rates of the terminal class that the customers belong to. And much like in the case of logistic regression above, these estimated response probabilities can be used to determine whether to include or exclude customers from a marketing campaign.

The main disadvantage of using a decision tree as a prediction model in the world of big data is the risk of over fitting, which require that one experiments with various parameters, termination rules and tree configurations to yield a stable tree with no over fitting. A remedy for the over-fitting issue may be provided by the random forest approach (Hastie et al. 2009) in which the forecast of the target variable is based on the average scores of dozens, or even hundreds, of decision trees which are built on multiple random samples drawn from the same training dataset. The idea being that if something is missed in one model, it will be captured in another model. Thus, predicted values based on the average of multiple scores are likely to yield more accurate predictions with less errors.

Alternatively, decision trees can be used for feature selection. In this case, one runs the decision classifier as part of the data preprocessing stage and then selects all attributes that took part in the splitting process as potential predictors for a predictive model, be that a linear regression model, a logistic regression model, or other. For example, in the decision tree above balance, residence-type, and age are reasonable candidates for a predictive model because they all took part in the tree-building process, whereas gender is excluded from this list of predictors because it did not participate in the splitting process.

11 Conclusions

Prediction models are assuming an increasing role in the world of big data for making decisions in many domains – marketing, banking, insurance, health care, cyber, and more. The complexity and dimensionality of the PA process and the huge number of potential predictors which could reach hundreds, if not thousands, render the model-building process very complicated and may increase the chances of over-fitting and non-stable models. The "wisdom" in building large-scale PA models is to account for all conflicting concerns and the trade-off between model accuracy and prediction errors.

In this article, we reviewed the process for building large-scale prediction models focusing on regression models, and to a lesser extent on decision trees, and discussed the differences between explanatory models and prediction models. We offered several performance measures for assessing the accuracy and prediction error for classification models based on logistic regression and the trade-off between them. We concluded the article with a short discussion on the deployment process of the predictive model and the manner the model results are applied for decision making in targeting problems.

References

Akaike, H. (1974). A New Look at the Statistical Identification Model, *IEEE Trans. Auto. Control*, 19, pp. 716-723.

Ben-Akiva, M., and Lerman, S.R. (1987). Discrete Choice Analysis, *the MIT Press*, Cambridge, MA.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing, *J. R. Statistical Society*, 57, pp. 289-300.

Bifet, A., Gavaldà, R., Holmes, G., and Pfahringer, B. (2018). Machine Learning for Data Streams with Practical Examples in Massive Online Analysis, *Adaptive Computation and Machine Learning*, MIT Press.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984), *Classification and Regression Trees*, Belmont, CA., Wadsworth.

DeGroot, M. H. (1993). *Probability and Statistics* 3rd edition, Addison-Wesley.

Diaconis, P. (1985). Theories of Data Analysis from Magical Thinking through Classical Statistics. In *Exploring data tables, trends and shapes* , Hoaglin, D. C., Mosteller, F. and Tukey, J. W. (edts), Wiley, NY, pp. 1–36.

Elkan, C. (1997). Boosting and Naïve Bayesian Learning, *Technical Report CS97-557*, University of San Diego, California.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural Networks and Bias/Variance Dilema, *Nueral Computation*, 4, pp. 1–58.

Efoymson, M.A. (1960). *Multiple Regression Analysis in Mathematical Method for Digital computers*, Wiley, NY, pp. 191–203.

Goldberg, D. E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning.* Addison-Wesley Publishing Company Inc.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of Statistical Learning,* Springer, NY.

Hochberg, Y., and Tamhane, A. C. (1987). *Multiple Comparison Proceedings*, John Wiley & Sons.

Kelleher, J. D., MacNamee, B., and D'Arcy, A. (2015). *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*, MIT Press.

Lambert, P. J. (1993). *The Distribution and Redistribution of Income,* Manchester University Press.

Levin, N., and Zahavi, J. (2005). Data Mining for Target Marketing, *The Data Mining and Knowledge Discovery Handbook*, Maimon O., and Rokach, L., (eds), Springer, NY, pp. 1261–1301.

Miller, A. J. (2002). *Subset Selection in Regression*, Chapman and Hall.

Murthy, K.S. (1998). Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey, *Data Mining and Knowledge Discovery*, 2, pp. 45–389.

Quinlan, J.R. (1986). Induction of Decision Trees, *Machine Learning*, 1, pp. 81–106.

Quinlan, J.R. (1993). *C4.5: Program for Machine Learning*, Morgan Kaufman Publishing, CA.

Rumelhart, D.E., McClelland, J.L., and Williams, R.J. (1986). Learning Internal Representation by Error Propagation, in *Parallel Distributed Processing: Exploring the Microstructure of Cognition*, Rumelhart, D.E., McClelland, J.L. and the PDP Researc Group, (eds.), MIT Press, Cambridge, MA.

Schwartz, G. (1978). Estimating the Dimension of a Model, *Annals of Statistics*, 6, pp. 486–494.

Shannon, C.E. (1948). A Mathematical Theory of Communications, *Bell System Technical Journal*, 27, pp. 379–423 & 623–656.

Shmueli, G. (2010). To Explain or to Predict? *Statistical Science*, 25, pp. 289–310.

Shmueli, G., and Koppius, O. (2011). Predictive Analytics in Information Systems Research, *MIS Quarterly*, 34, pp. 553–572.

Tukey, J.W. (1977). Some Thoughts on Clinical Trials, Especially Problems of Multiplicity, *Science*, 198, pp.679–684.

Vapnik, V.N. (1995). Support Vector Machines, *Machine Learning*, 20, pp. 273–297.

Machine Learning for the Geosciences

Neta Rabin and Yuri Bregman

1 Introduction

Geoscience is a branch of science that focuses on studying the Earth's subsystems. These include studying the Earth's soil, metallic core, surface, and interior. It is also concerned with investigating the oceans, rivers, lakes, ice sheets, glaciers, and the atmosphere. The analysis relies on model-based simulations, algorithms, and data-driven analysis. The increasing available amounts of recorded data, which come in the form of images, maps, tables, and continuous waveforms, to name some, pose some challenges such as data integration and massive data processing.

Machine learning is now rapidly expanding in all science and engineering domains that process big data. In geoscience, a large number of research papers utilize machine learning techniques for problems that were traditionally solved with other techniques. Several recent review papers [1, 2, 3, 4] describe central machine learning methods and their applications to different geoscience domains. Some of these papers approach the geoscience research community, as they aim to engage scientists in the field of machine learning. Other papers reach out to the machine learning community and explain the current challenges in geoscience. In this review paper, we approach researchers from the machine learning community and describe several data-driven fields in geoscience, with a special focus on seismology.

There are a variety of subfields in the geosciences that study various features of the Earth and its dynamic systems. Apart from the common goal of studying the Earth, the geoscience sub-fields all share the theme of learning from observations.

N. Rabin (✉)
Tel-Aviv University, Tel-Aviv, Israel
e-mail: netara@tauex.tau.ac.il

Y. Bregman
Soreq Nuclear Research Center, Yavne, Israel
e-mail: yuri@soreq.gov.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_34

Thus, as the size and quality of gathered data increase, data-driven modeling naturally becomes a significant form for understanding and prediction of the Earth's phenomena. Below, we mention several sub-fields in geoscience, in which machine learning algorithms are presently implemented.

Geology is centered on learning the Earth's materials, organisms, the Earth's structure, and how it has changed overtime. The need to move from knowledge-driven analysis to data-driven analysis in the field of resource estimations, and in particular for determining the presence of gold in rock, is discussed in [5]. Application of random forests from different data sources that are gathered for mining explorations is described in [6]. Moreover, random forests are used for estimating sodium, a key geochemical element, by analyzing a multivariate chemo-physical dataset measured on drill cores. There is a long and rich history related to inverse problems and simulation analysis in the oil and gas field [7]. A recent paper [8] deals with characterization of oil reservoirs based on geological properties and data extracted from seismic recording. Multi-gene genetic programming, which enables to model nonlinear relationships between variables, was applied for estimating porosity in an oil reservoir.

Lithology, which is a branch in geology that studies the physical characteristics of rocks, often relies on remote sensing techniques. These aim to find the underlying physical characteristics of the studied area by monitoring and measuring from a relatively far distance the reflected and emitted radiation from the area of interest. The measured data are typically high-dimensional; therefore, machine learning methods may simplify the analysis. Several machine learning methods were applied in [9] for recognizing patterns in data collected by airborne geophysics and multispectral satellite for the task of lithology classification. These detailed examples illustrate some of the challenges in geology, where data are gathered by multiple sensors of different types and inverse problems are aimed to be solved. Integration of machine learning algorithms that support sensor-fusion-based modeling opens a window of opportunities for making new discoveries by processing the large amounts of the various gathered datasets.

The field of hydrology studies the Earth's waters, which is a valuable resource. Nowadays, scientific research is aimed at monitoring and development of reliable prediction algorithms for tracking water movement, planning water usage, and assuring water quality. Many resources are invested for advancing the data acquisition technologies that support the water industry and water research. For example, the gravity recovery and climate experiment (GRACE) [10], which was a joint mission of NASA and the German Aerospace Center, sent satellites that provided valuable data for tracking water storage dynamics. Sun et al. [11] developed deep convolutional neural network models for learning spatial and temporal patterns of mismatch between total water storage anomalies that were recorded from GRACE and simulated by a widely used land surface model. Prediction of rain and snow rates in high latitudes based on satellite remote sensing data from multiple sources was processed in [12] by application of deep neural networks. The results show that the deep learning model outperforms prevalent rain prediction models. Long-term prediction of water flow and level in rivers was addressed in [13] by utilizing

deep convolutional neural networks, which tackled the challenge of data fusion. Such tasks are useful for water authorities, who need to plan the allocation of water resources for agriculture and for domestic use. Another important task is the monitoring of water quality, which is under the responsibility of reservoir managements. In [14], an approach that evokes four machine learning models (artificial neural networks, support vector machines, classification and regression trees, and linear regression) was established for predicting a known measure for water quality, the Carlson's Trophic State Index [15]. Such an approach can assist water managers and engineer in assuring water quality, while reducing the cost and time of conducting chemical experiments. Overall, the data-driven challenges in the field of hydrology include data fusion, processing of high-dimensional remote sensing data, time-series prediction, and development of reliable models for helping us to wisely manage the water system.

Climate modeling, in which the dynamical behaviors of the atmosphere, oceans, and land are coded into mathematical models, also incorporates machine learning method to assist in modeling some of the complex nonlinear dynamical phenomena. A machine learning toolbox name ClimateLearn was developed by Feng et. al. [16] for climate prediction task, which also includes data merging and data cleaning solutions. Prediction of sea ice anomalies based on analog forecasting with dynamics-adapted similarity kernel was presented in [17]. This forecasting problem was classically modeled by using single historical records. Machine learning tools can better learn the variability of the sea surface temperature by using ensembles of analogs for forecasting. A deep learning model was applied in [18] to overcome the course nature of simulation models, which are limited due to computational complexity. The data-driven deep learning model allows fast and accurate short-term prediction of cloud-resolving simulations. Another paper that investigated neural networks as an alternative to numerical simulation-based predictions of climate is [19]. The dynamics of the Portable University Model of the Atmosphere (PUMA) [20], which is a simple general circulation model, was learned by the network in the training step. Then, the network was used for making future weather prediction. Last, in [21], the authors review several machine-learning-based models for prediction of the El-Niño climate pattern.

As reviewed above, the processing obstacles in geoscience are related to large amounts of varied data and complex dynamical phenomena that are computationally demanding on the one hand, but required for detailed forecasting and prediction on the other hand. Techniques and tools from the field of machine learning are now being implemented, and they close some gaps by providing sensor fusion models and methods that result in more accurate data-driven simulations and future predictions of dynamic, complex systems.

In what follows, we show the application of machine learning methods to the geoscience focusing on seismology, which is another sub-field of geoscience. In Sect. 2, basic seismological terms, concepts, and processing techniques are presented. Section 3 contains a schematic diagram that describes commonly used steps in machine-learning-based algorithms for seismology. Description of the detection and classification tasks together with examples of machine learning model

implementations is given in Sects. 4 and 5, respectively. Recent applications of deep learning methods to event detection and classification tasks are reviewed in Sect. 6. Last, conclusions are presented in Sect. 7.

2 Background on Seismology

Seismology is the study of earthquakes and seismic waves that move through and around the Earth. The field also includes studies of earthquake environmental effects such as tsunamis as well as diverse seismic sources such as volcanic, tectonic, oceanic, atmospheric, man-made explosion processes, and earthquake forecasting. One of the key problems in seismology is to solve the inverse problem, i.e., to derive from the analysis of seismic records information about the structure and physical properties of the Earth medium. Seismology is driven by observations; thus, improvements in instrumentation and data availability had always contributed to the progress in seismology theory and in our understanding of the Earth's structure. There are several different kinds of seismic waves, and they all move in different ways and with different velocities [22]. The two main types of waves are body waves and surface waves. A body wave is a seismic wave that moves through the interior of the Earth, as opposed to surface waves that travel near the Earth's surface. There are two types of body waves, pressure waves or primary waves (P-waves) and shear or secondary waves (S-waves).

Seismic waves are recorded by seismometers that measure the motion of the ground. Most modern seismometers include three separate channels that allow the determination of the simultaneous movement in three different directions: up–down, north–south, and east–west. A typical seismic station consists of a three-component (3-channel) seismometer, a GPS clock for determining time, and a recorder for collecting data [23]. A recording of the Earth motion as a function of time is called a seismogram (a seismic waveform). S-waves are slower than P-waves. Therefore, in the seismograms, the recording of S-waves (S-phase) appears later than P-phase. Some of the seismic stations were equipped by seismic arrays, which improve the signal-to-noise ratio (SNR) by finding directional information [24].

Seismic data processing includes seismic signal detection and phase picking, calculation of the origin time and location of the hypocenter (i.e., rupture starting point), and seismic event characterization (estimation of magnitudes and event classification) [25]. The signal detection and the phase picking are performed using the waveforms of each station separately, while the event location and characterization are network-level processing tasks, which involve multiple station processing. A significant step of event location is seismic phase association, i.e., the task of linking together phase detections on different seismometers that originate from a common seismic source. The analysis results are compiled in a seismic bulletin, i.e., a catalog including the hypocenter parameters together with the attributes of all detected phases (i.e., station, phase arrival time, amplitude, etc.).

An important development in automatic seismic detection was the STA/LTA trigger-based algorithm [26, 27, 28], introduced in the late 1970s. It utilizes as a criterion for picking the ratio of continuously calculated average energy (or envelope or absolute amplitude) of a recorded trace in two consecutive moving-time windows of different lengths. The first calculates a short-time average (STA), which captures the signal energy, and the second calculates a long-term average (LTA) that estimates current energy of the noise.

A large number of machine learning papers that present applications to seismic event identification and classification rely on computing a time–frequency representation of the signal as a first phase and processing it to extract features as a second step. The processed time–frequency representations are then fed as input to a machine learning algorithm. In Sect. 3, we provide a general block diagram, which explains the workflow of many algorithms that were developed for seismic processing with machine learning, in the spirit of the SonoDet detector.

Today, the analysis tasks in most seismic observatories start with near real-time automatic processing that implements automatic signal detection and P-phase picking, hypocenter's localization, and magnitude determination. However, the current quality of most automatic seismic bulletins is insufficient. Therefore, seismological institutes heavily rely on manual analysis for performing routine tasks. Seismologists interactively revise and correct the automatic bulletins based on scientific reasoning, their knowledge, and practical experience.

3 A General Machine Learning Framework for Seismic Signal Processing

Many of the developed machine learning methods for seismic analysis rely on a set of similar processing steps. A general schematic block chart, which describes the central steps that are applied in machine learning algorithms for seismic signal analysis, is displayed in Fig. 1. These steps are also typical for other machine learning signal processing applications, and the main difference is the feature extraction step, which may rely on seismological expert knowledge.

The left box describes the input data, which contain waveforms from a single seismic channel, from multiple channels, or from a seismic array. Then, an optional prepossessing of the raw data may be applied. This step can include signal filtering

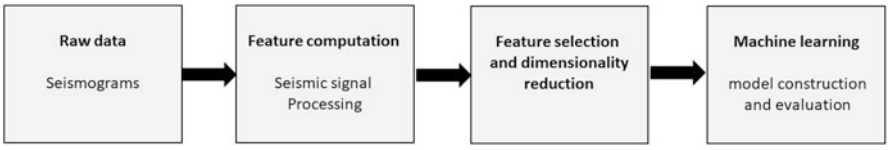


Fig. 1 Schematic description of seismic signal processing followed by machine learning steps

or sensor fusion computations such as beamforming algorithms [24] for array data. Next, feature computation is performed. In this step, the rich historic experience of seismic signal processing comes in, and features that rely on theoretical understanding of the studied problem are computed. This step results in a finite set of features computed from the time and frequency domain of each seismogram. Alternatively, a unified time–frequency representation, named a spectrogram, can be calculated and used as the seismogram's features. The feature extraction step is typically followed by algorithms that perform signal selection or dimensionality reduction. These simplify the feature space by reducing the number of inputs that are fed into the last algorithmic box. The machine learning model is constructed from a subset of the data, a training set, which is represented by the previously calculated features. For a classification problem, the seismic event types of the training set are known. This knowledge is based on seismic bulletins and manual classifications. Last, the machine learning model is evaluated and applied to the test data for performing the task at hand. In fact, this scheme is very similar to application of machine learning methods in other fields of signal processing using features that were utilized by the seismologists throughout the years.

A detailed example that follows the general model presented in Fig. 1 is described. A machine learning model for earthquake–explosion discrimination was presented in [29]. The raw data included seismogram waveforms of events from the Dead Sea area, recorded from two stations with three channels per station.

The feature extraction step constructed a time–frequency representation from the given waveforms of each channel in each station. It was based on sonograms method [30, 31] with several modifications. A single-trace seismic waveform was decomposed into a set of overlapping windows. Then, the short-time Fourier transform is applied to the waveform segment in each time window, which was preliminary Hanning-tapered, and power spectral densities were computed. The sonogram was obtained by summing the spectrogram in logarithmically scaled frequency passbands for every time bin. The frequency scale is rearranged to be equally tempered on a logarithmic scale into near half-octave logarithmically scaled frequency passbands. Finally, the sonogram is normalized such that the sum of energy in every frequency band is equal to 1 and the result is a normalized sonogram. A similar normalization of sonograms was used in [32] for volcanic seismic signal classification and in [33, 34] for identification of repeated events.

Figure 2 presents four sample seismograms and their corresponding normalized sonogram feature representation. The left part displays four seismograms that were recorded at the vertical channel of the MMLI station. The first three (from top) seismograms belong to earthquakes in the northern Dead Sea area with duration magnitudes Md of 5.1, 2.6, and 3.5, respectively, and the bottom seismogram belongs to a co-located underwater explosion with duration magnitude Md =3. The right part presents the resulting normalized sonograms. Although the original seismograms have huge differences in amplitude ranges (between 10 and 10^7), the normalized sonograms transform all waveforms in similar ranges of amplitudes.

The third step of Fig. 1 is performed in this example by the application of a nonlinear dimensionality reduction method named diffusion maps [35]. Diffusion

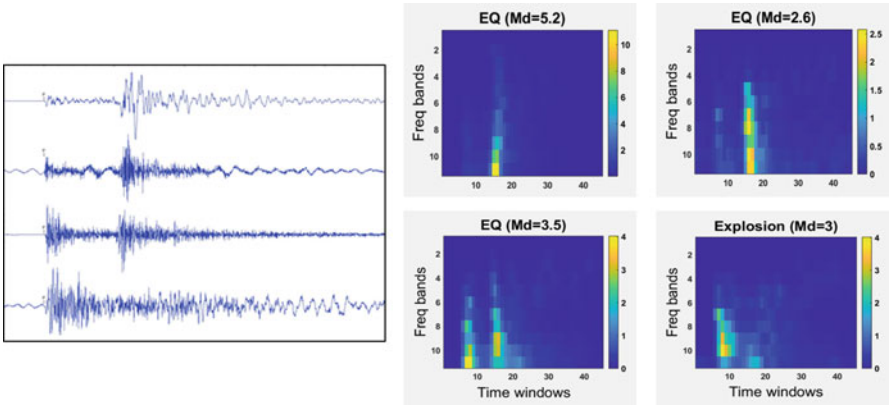


Fig. 2 Left: Four seismograms: earthquakes with duration magnitudes Md of 5.1, 2.6 and 3.5 and an underwater explosion with Md = 3, ordered from top to bottom. Right: The resulting normalized sonograms of the four seismograms (Figure taken from [29])

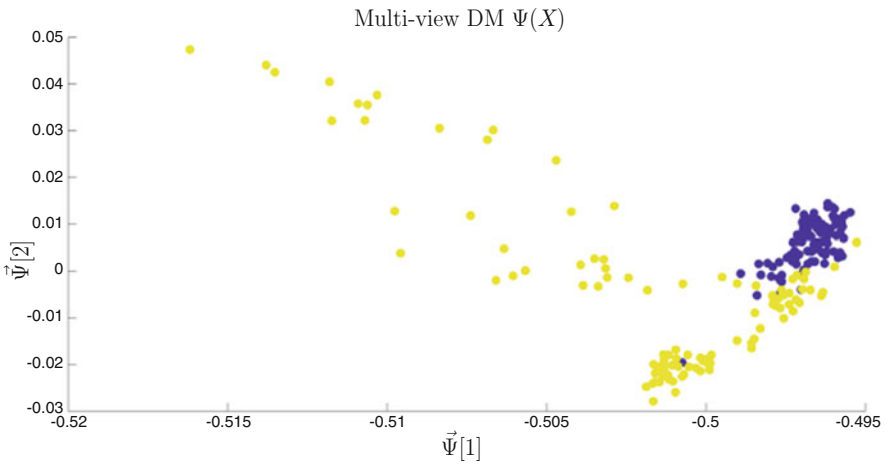


Fig. 3 Two-dimensional multi-view diffusion maps mapping. Blue points represent explosions, and yellow points represent recordings of earthquakes (figure taken from [36])

maps take the set of computed sonograms as input and reduce the dimension of each such sonogram. When a new test seismogram arrives, it undergoes the feature extraction steps and its sonogram is computed. Then, it is embedded in the low-dimensional space. The final classification step (which is associated with the bottom-left box in Fig. 1) is performed with k-nearest neighbors. A modification of the diffusion maps algorithm, multi-view diffusion maps, which perform sensor fusion and dimensionality reduction, was presented in [36] and plotted in Fig. 3. This resulted in an improved low-dimensional representation of seismograms that were recorded from multiple channels.

4 Seismic Event Detection and Localization

Seismic event detection is the real-time detection of seismic event from a continuous seismic recording. The prevalent techniques are trigger-based algorithms that compute amplitude thresholds or implement a pattern recognition method [26, 27]. Other trigger-based algorithms, which extend the STA/LTA approach, are described in [37]. Detectors that are based on higher-order moments have been proposed in [38, 39, 40] for determining the onset of seismic waves. A nonlinear dimensionality reduction method was applied for the estimation of arrival times in [41] and for the seismic phase classification in [42].

In order to improve the seismic signal processing capabilities for detection and the classification problems, similarity-based algorithms were developed. The main transition was the advancement from signal processing methods, which analyze each event separately, to methods that learn from historic data. The earlier methods focused on template-based algorithms, and they developed in two directions: pattern recognition techniques and correlation-based analysis.

The template-based detection utilizes the long-known observation that seismic events from close locations, with similar source mechanisms, recorded by the same receiver, often generate very similar seismograms while displaying different amplitudes [47, 48]. This is because the combination of the source mechanism and the path from source to receiver effectively determine the received signal at a given station [25]. Template-based approach is especially suited for the seismicity occurring in a circumscribed area with similar source mechanism (the so-called repeating events) like aftershocks and mining explosions. For instance, this approach enables to select the seismograms of strong aftershocks with high signal-to-noise ratio and subsequently to utilize those templates to detect the weak aftershocks with low signal-to-noise ratio.

One of the first template-based methods was the SonoDet detector, initially developed to monitor induced seismicity in Germany [30, 31]. This method implements the STA/LTA detector with a very low threshold as a preprocessor and subsequently performs the identification step using the smoothed time–frequency representation of a single-channel seismogram, named a sonogram. SonoDet detector and its later variants have found various applications that included detection and classification of seismic signal with very low signal-to-noise ratio [43, 44, 45, 46].

From the early 1990s to this day, template-based methods named waveform cross-correlation detectors are developed. These methods are vastly used nowadays as well. Waveform cross-correlation techniques may be applied for event detection, localization, and even characterization of repeated seismic events including aftershock sequences [49, 50, 51, 52, 53, 54]. Although this does not fall into the category of machine learning algorithms, the idea of pattern recognition learning is presented and resembles processes used in machine learning.

A drawback of waveform cross-correlation detectors is a relatively high false alarm rate. The false alarms originate not only from detection of noise but rather from seismic arrivals with unrelated source locations. In [55], the authors

presented a new approach to waveform correlation that utilizes techniques from supervised machine learning such as support vector machines (SVMs). Waveform templates were generated from a large labeled waveform training set of known historical seismic arrivals and used to discriminate between arrivals from a specific geolocation source and all other arrivals. Apparently, this is the first time that machine learning has been applied to generate alternative templates for seismic event detection.

An alternative unsupervised pattern-mining approach, named Fingerprint and Similarity Thresholding (FAST), was introduced in [56] and extended in [57, 58, 59]. Following Fig. 1, the method starts with waveform from a single channel. Feature extraction is denoted by fingerprint extractions, which is a modification of the Waveprint method developed for audio identification [60]. In the feature extraction step, spectrograms followed by Haar wavelet transforms are computed. These result in spectral images that represent the studied seismogram. Then, the significant image coefficients are selected and coded into a compact binary fingerprint scheme. A locality-sensitive Hashing [61] next groups together similar fingerprints. Last, detection of new events is performed by a similarity search, which is efficient because of the hashing procedure. FAST is computationally more efficient than template-based detection methods and can process up to 10 years of continuous datasets [62].

The Comprehensive Nuclear-Test-Ban Treaty organization (CTBTO) implemented the Network–Vertically Integrated Seismic Analysis (NET–VISA) for global nuclear explosion monitoring [67, 68]. The system adapted a Bayesian framework with a forward physical model using probabilistic representations of the propagation, station capabilities, background seismicity, and noise statistics to obtain the maximum a posteriori solution to the nonlinear problems of phase association and event location. Later the core seismic model was supplemented with a model of underwater as well as atmospheric events. The model is regularly retrained on many months of historical data to fine-tune the priors and the likelihoods. So, the system is able to detect and locate seismic, hydro-acoustic, and infrasound events considering the cross-over of energy between different mediums. NET-VISA is a fully fledged operational machine learning-based system running at the CTBTO since 2017. The latest event detection methods apply deep learning techniques, and some of these are reviewed in Sect. 6.

5 Seismic Event Classification

Typical event classification algorithms (also known as event discrimination) calculate several seismic parameters from the input seismograms and use these parameters to distinguish between earthquakes and explosions. The simplest seismic parameter is focal depth. Its drawback is that estimation of focal depth is usually inaccurate in lack of depth phases. Other widely used seismic discrimination methods are Ms:mb (surface wave magnitude versus body wave magnitude) and

spectral amplitude ratios of different seismic phases [63, 64]. However, discrimination methods based on seismic parameters give only a partial solution to the discrimination problem. Due to these misclassifications, regional earthquake catalogues are often contaminated with explosions, which may cause the erroneous estimation of a seismicity hazard [65].

Over the last two decades, different machine learning algorithms were applied to classify seismic events in an unsupervised or in a supervised mode. Classification tasks that are related to separation of the events into two classes: natural and man-made events are known as discrimination problems. This task is often related to monitoring and discrimination of nuclear explosions [66].

Several machine learning methods were utilized for the seismic discrimination problem. Artificial neural networks were applied in [69] after a feature extraction step that included complexity, spectral ratio, and third moment of frequency. In [70], the feature extraction step was based on Autoregressive Moving Average (ARMA) coefficient filters, and artificial neural networks were applied for discriminating between earthquakes and man-made events. Neural networks for discrimination were also applied in [71, 72, 73, 74], in [75] for discriminating between deep and shallow earthquakes, and in [76] for discriminating between local, regional, and teleseismic earthquakes.

Kuyuk et al. [77] used self-organizing maps and artificial neural networks combined with unsupervised learning for discrimination between small earthquakes and quarry blasts. Self-organization maps [78] were additionally applied for automatic recognition of volcano-seismic signal patterns in [79] and for single-station classification of seismic events in [80] by application of principal component analysis [81] for dimensionality reduction and self-organization maps for classification. Reynen and Audet [82] have applied a logistic regression classifier for earthquake–explosion discrimination of events detected at a network of 13 three-component stations in Southern California. Using frequency-averaged spectrograms and polarization features, they reported classification accuracy of over 99%.

Support vector machine (SVM) [83] is another common machine learning technique that was applied for seismic event classification to data collected from multiple stations in Finland ([65]). The feature extraction step included computation of 80 features for each input waveform and resulted in a 94% correct classification rate for the non-earthquakes and a correct classification of all but one of the earthquakes. Feature extraction using the wavelet transform followed by SVM was suggested in [84, 85] for discriminating between earthquakes and quarry blasts. SVM applied after a kernel-PCA dimensionality reduction was discussed in [86] and demonstrated for classification of synthetic waveforms. Two unsupervised machine learning methods, k-means and Gaussian mixture model, were applied for classification of seismic activities in Istanbul [96].

Hidden Markov model (HMM)-based earthquake detection and classification method is another notable automatic learning approach. The main algorithm was borrowed from speech recognition. This approach is originally introduced for the classification of seismic signals of volcanic origin [32] allowing to learn classifier properties from a single waveform example and some hours of background

recording. Later the modifications of this method were applied for the detection and distance-dependent classification of small earthquakes recorded by the Bavarian Earthquake Service [87], volcano signal classification [88, 89, 90], classification between earthquakes, explosions and rockfalls [93], geothermal reservoir monitoring [91], and array data classification [92]. Dynamic Bayesian networks, which generalize HMMs, have been used for real-time classification of seismic signals in context of supervised learning [94, 95]. Two unsupervised machine learning methods, k-means and Gaussian mixture model, were applied for classification of seismic activities in Istanbul [96].

Another impotent seismic processing task is concerned with earthquake early warning (EEW) systems, which are aimed to detect significant earthquakes so quickly that alerts can reach many people before shaking arrives. The developed method must rapidly estimate magnitudes and discriminate is between earthquakes and mine and quarry explosions given a few seconds after the arrival of the P-wave. MyShake is one of the interesting current earthquake early warning projects [97]. Its goal is to build an EEW system based on a crowd-sourcing global smartphone seismic network. Analyzing seismic data from smartphones require complex tasks that benefit from various machine learning tools including the artificial neural network, the density-based spatial clustering of applications with noise (DBSCAN), random forests, and convolutional neural networks (CNNs) [98]. Recent advances in machine learning for early warning that rely on deep learning techniques are referred to in Sect. 6.

6 Deep Learning for Seismic Signal Processing

The use of deep learning techniques in the field of geoscience and in particular seismology is relatively new, and most papers were published after 2017. A large number of papers focus on the problem of event detection and phase association. Deep convolutional networks were proposed for this task in a number of papers. The input to networks is either the seismograms or a time–frequency representation of them. Later papers utilize other deep network models such as residual, recurrent, and generative adversarial neural networks. Alongside, deep learning models have been explored for improving other seismic tasks such as hypocenter's localization, event characterizations, and more general problems such as signal denoising.

We review a number of recent papers that utilize deep convolutional networks and other deep learning methods for seismic event processing. For each work, we mention the type and size of training set that was used and provide details about the models' performances.

6.1 Seismic Event Detection Using Deep Learning Techniques

A deep convolutional network for performing earthquake detection and location based on a single waveform was proposed in [99], denoted as ConvNetQuake. The model's input is a window of three-channel seismograms data (the raw time-series data), and each window consists of 1000 samples. The training set contains 2,709 windows including seismic events and 700,039 noise windows. The output is a predicted label, of either a seismic event or seismic noise. Each convolutional layer has in it 32 filters that down-sample the data by a factor of 2. Eight convolution layers are required to flatten the data into a 1D vector of 128 features. A fully connected layer outputs the predicted class scores. The test set performance of ConvNetQuake has demonstrated a 94.8% precision (fraction of detected events that are true events) and a 100% recall (fraction of true events correctly detected). Based on the ConvNetQuake model, the deep convolutional network ConvNetQuake_INGV was investigated in [100]. The network aims to detect and characterize global earthquakes over a broad range of epicentral distances and magnitudes. These include characterizing the earthquake's parameters such as the distance, azimuth, depth, and magnitude of the event. The training set included 15,200 events and 10,724 noise waveforms recorded at different three-component stations. The detection precision of ConvNetQuake_INGV on the test set is 97% and recall is 81%, while it shows moderate overall performance for predicting event distance, azimuth, depth, and magnitude for the test dataset.

An additional deep learning application for the problem of earthquake early warning is described in [101]. A combination of generative adversarial networks (GANs) and random forests was used. The GAN network was inserted for distinguishing between real earthquakes and noise signals to reduce false alerts. The GAN network was trained on $300,000$ waveforms recorded in southern California and Japan. Then, the output from the GAN networks, which includes produced synthetic samples, is used for training a random forest classifier with about 700,000 earthquake and noise waveforms. The high-discrimination results show that 99.2% of the earthquake P-waves and 98.4% of the noise signals can be correctly recognized by the combined method. Another model that is proposed for the problem of earthquake early warning in [102] is a long short-term memory network (LSTM), applied to data gathered in Japan. The input to the LSTM network was based on computed features from the short 1 s P-wave recordings. The method's accuracy is above 98%.

Separating the seismic signal from noise is another task related to earthquake detection. This topic was addressed in [103] by developing a deep-learning-based denoising algorithm named DeepDenoiser. The convolutional neural network learns a sparse representation of data that is used for decomposing the input signal into two parts: a signal of interest and noise. This type of decomposition is shown to improve earthquake detection, and it may be useful as a prepossessing step for a variety of seismic learning tasks.

In [104], the authors used a CNN with three convolutional layers for P-wave arrival picking and first-motion polarity discrimination. The networks were trained on 18.2 million seismograms recorded by the Southern California Earthquake Data Center that included with 4,847,248 manually determined P-wave picks. Through cross-validation on 1.2 million independent seismograms, the differences between the automated and manual picks showed to have a standard deviation of 0.023 secs.

Zhu and Beroza [105] introduced the PhaseNet, a U-shaped deep CNN [106] that picks the arrival times of both P- and S-waves. The network was trained on over 7,000,000 manually labeled seismic records from over thirty years of earthquake recordings from the Northern California Earthquake Data Center. PhaseNet uses three-component seismic waveforms as input and generates probability distributions of P-arrivals, S-arrivals, and noise as output. They demonstrated that this deep learning method achieves much higher picking accuracy and recall rate than existing methods.

The application of CNNs on relatively smaller sized training sets for seismic phase classification was tested in [107]. Around $11,000$ of P- and S-phase pair were used as input to a CNN to classify seismic phases. Three one-dimensional waveform sample windows, which correspond to 3 components, were fed to the network. The network outputs the probability of P-phase, S-phase, or noise for each time sample within that window. A softmax or normalized exponential function was used in the final layer for generating the associated probabilities for each class. The experimental results show that even when data are scarce, a CNN architecture significantly improves the P- and S-arrival pick residuals. When comparing the results to a classical approach, optimized short-term average/long-term average (STA/LTA) [108], the CNN outperforms the STA/LTA approach and achieves results that are close to those done by manual inspection.

The phase association task was recently addressed in [109], where a method named PhaseLink was introduced. PhaseLink is a deep learning approach that utilizes recurrent neural networks (RNNs) for learning temporal and contextual relationships in sequential data. Another advantage of this method is that it is trained on synthesized data samples generated from simple 1D velocity models. The training set included about six billion artifactually generated seismic phase arrival times. So, even in the most seismically active regions, the available data may barely be enough to effectively train such a deep network. PhaseLink was applied to associate more than 70,000 arrival times that were picked by seismic analysts and has demonstrated the state-of-the-art performance (precision and recall were 0.98 and 0.96, respectively). Phase association for pairing between waveform arrival from two stations for predicting whether they originate from a common source was investigated in [110], while convolutional neural networks were applied to a large dataset from Chile.

6.2 Seismic Event Localization and Characterization with Deep Learning Models

Seismic event localization is another common task that is used for determining the location of a seismic event. Traditional methods use the arrival times of seismic phases in different stations to estimate the event's location. Deep learning was recently utilized for event localization in [111] by implementing a deep CNN applied to a small area in West Bohemia. The network was trained on 2118 localized earthquakes from the same region. Input consisted of three-component full-waveform records of multiple stations. The network was evaluated on 908 events, which were located with small standard deviation errors of 56.4 m in east–west, 123.8 m in north–south, and 136.3 m in vertical direction.

Seismic event characterization tasks have also been approached with deep learning techniques. CNNs were constructed in [112] for discriminating between tectonic tremor, local earthquakes, and noise. The input to this network was a set of 17,213 spectral images of size 64×64 pixel, which capture the event's high-resolution features in both frequency and time domains. The constructed CNN was sensitive to the absolute frequency of signal appearance, but at the same time invariant to the time of the signal onset. An accuracy of 99.5% was achieved.

In [113], convolutional and recurrent neural networks were used to accomplish discrimination of earthquakes and explosions for local distances. Using a 5-year manually reviewed event catalog generated by the University of Utah Seismograph Stations (13,313 events; 103,944 phases), the authors train the deep learning models to produce automated event labels using 90 sec. event spectrograms from three-component and single-channel sensors. Both network architectures are able to replicate analyst labels 98% of the time. Moreover, each model is able to identify human errors within the event catalog (label noise constitutes about 1% of the catalog).

The classification performance of CNN model from previous work [113] was compared in [114] with a traditional, physics-based spectral amplitude ratio method. The test dataset contained a manually reviewed local catalog of seismic events recorded during a 14-day period by the same network of University of Utah (7,377 events). The task was to provide more detailed classification than earthquake–explosion discrimination, for instance, attributing explosions to known mines. The CNN model has achieved success rates between approximately 91% and 98% versus rates of 80% to 90% for the amplitude ratio method. Thus, the deep learning approach outperforms the traditional method, and a major advantage of CNN is its robustness to low signal-to-noise-ratio data, allowing to classify significantly smaller events.

In [115], the authors introduce a seismic event discrimination model based on deep canonical correlation analysis [116], which extracts a nonlinear transformation from sonograms into a low-dimensional space. It is shown that even based on relatively small sample size (1,609 events), the neural net-based representation (DCCA) outperforms the state-of-the-art kernel-based methods.

7 Conclusions

Machine learning methods are nowadays used in many sub-fields of geoscience for performing data-driven analysis, accompanying traditional signal processing, hypothesis-based analysis, and numerical simulation models. In this review paper, we described several machine learning applications to geoscience with a special focus on observational seismology. Nonetheless, several interesting applications, even in the seismology, have not been discussed in detail. Topics such as simulation of seismic waves and full-waveform inversion [117, 118, 119, 120], seismic hazard assessment [121, 122, 123], and prediction of experimental laboratory earthquakes [124, 125] have been approached by machine learning and deep learning techniques over the last years. Earthquake prediction attracted the interest of hundreds of teams during the "earthquake prediction" competition on the Kaggle platform (Los Alamos National Laboratory, 2019 [126]). Machine learning methods including neural and deep networks were applied to earthquake prediction. So far, machine learning predicts well the timing and size of laboratory earthquakes by reconstructing and properly interpreting the spatio-temporal complex loading history of the system. Although these results promise substantial progress in real earthquake forecasting, machine learning has not yet led to a breakthrough in earthquake predictability [127].

The majority of research papers describe the use of machine learning techniques for supervised problems. However, since geoscience is driven by underlying physical phenomena, utilization of advanced learning techniques, which model the data with respect to the underlying intrinsic geophysical phenomena, may also be an exciting future direction for forming data-driven models.

One drawback of some existing machine learning and deep learning techniques that are applied in geophysics is the large dependency on the training set. There is no guarantee that a model that was constructed using data gathered form one area would work as well on a different dataset. In addition, since many of the deep learning models require huge amounts of labeled data, these types of algorithms may not be applicable to regions that are sparse in data, for example, in seismology, for low-seismicity regions. Another known drawback of deep learning and some machine learning algorithms is their "black box" nature, which makes it hard to interpret the underlying features that drive the classification results. One example is the field of nuclear test monitoring, where geophysical results may have significant political consequences.

On the other hand, machine learning and deep learning methods often outperform other detection and classification techniques; therefore, they are suitable for tasks such as earthquake early warning, where rapid and accurate detection is required. In addition, machine learning methods can improve our understanding of the physical phenomena. Moreover, the rapid development in this field encourages researchers to collect labeled data of good quality and to make it available to a large community of developers. Last, it is expected that future mutual fertilization between machine learning and geoscience will advance both fields, by providing real-life, complex

challenges for the machine learning community and by relying on data-driven analysis (to enhance or replace model-based techniques) in the geoscience field.

References

1. Karpatne, A., Ebert-Uphoff, I., Ravela, S., Babaie, H. A., Kumar, V.: Machine learning for the geosciences: Challenges and opportunities. IEEE Transactions on Knowledge and Data Engineering, **31**(8), 1544–1554 (2018).
2. Lary, D. J., Alavi, A. H., Gandomi, A. H., Walker, A. L.: (2016). Machine learning in geosciences and remote sensing. Geoscience Frontiers, **7**(1), 3–10 (2016).
3. Bergen, K. J., Johnson, P. A., Maarten, V., Beroza, G. C.: Machine learning for data-driven discovery in solid Earth geoscience. Science, **363**(6433), eaau0323 (2019).
4. Lary, D. J., Zewdie, G. K., Liu, X., Wu, D., Levetin, E., Allee, R. J., Malakar, N., Walker, A., Mussa, H., Mannino, A., Aurin, D.: Machine learning applications for Earth observation. Earth observation Open Science and Innovation, **165** (2018).
5. Caté, A., Perozzi, L., Gloaguen, E., Blouin, M.: Machine learning as a tool for geologists. The Leading Edge, **36**(3), 215–219 (2017).
6. Schnitzler, N., Ross, P. S., Gloaguen, E.: Using machine learning to estimate a key missing geochemical variable in mining exploration: Application of the random forest algorithm to multi-sensor core logging data. Journal of Geochemical Exploration, **205**, 106344 (2019).
7. Bedrikovetsky, P.: Mathematical theory of oil and gas recovery: with applications to ex-USSR oil and gas fields, **4**, Springer Science & Business Media (2013).
8. Maynard, J. A., Talavera, A., Forero, L., Pacheco, M. A. C.: Estimating the Geological Properties in Oil Reservoirs Through Multi-Gene Genetic Programming. In 2018 IEEE Congress on Evolutionary Computation (CEC), 1–5 (2018).
9. Cracknell, M. J., Reading, A. M.: Geological mapping using remote sensing data: A comparison of five machine learning algorithms, their response to variations in the spatial distribution of training data and the use of explicit spatial information. Computers & Geosciences, **63**, 22–33 (2014).
10. Tapley, B. D., Bettadpur, S., Watkins, M., Reigber, C.: The gravity recovery and climate experiment: Mission overview and early results. Geophysical Research Letters, **31**(9) (2004).
11. Sun, A. Y., Scanlon, B. R., Zhang, Z., Walling, D., Bhanja, S. N., Mukherjee, A., Zhong, Z.: Combining Physically Based Modeling and Deep Learning for Fusing GRACE Satellite Data: Can We Learn from Mismatch?. Water Resources Research, **55**(2), 1179–1195 (2019).
12. Tang, G., Long, D., Behrangi, A., Wang, C., Hong, Y.: Exploring deep neural networks to retrieve rain and snow in high latitudes using multisensor and reanalysis data. Water Resources Research, **54**(10), 8253–8278 (2018).
13. Assem, H., Ghariba, S., Makrai, G., Johnston, P., Gill, L., Pilla, F.: Urban water flow and water level prediction based on deep learning. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 317–329 (2017).
14. Chou, J. S., Ho, C. C., Hoang, H. S.: Determining quality of water in reservoir using machine learning. Ecological Informatics, **44**, 57–75 (2018).
15. Carlson, R. E.: A trophic state index for lakes 1. Limnology and Oceanography, **22**(2), 361–369 (1977).
16. Feng, Q. Y., Vasile, R., Segond, M., Gozolchiani, A., Wang, Y., Abel, M., Havlin, S., Bunde, A., Dijkstra, H. A.: ClimateLearn: A machine-learning approach for climate prediction using network measures. Geoscientific Model Development, 1–18 (2016).
17. Comeau, D., Giannakis, D., Zhao, Z., Majda, A. J.: Predicting regional and pan-Arctic sea ice anomalies with kernel analog forecasting. Climate Dynamics, **52**(9–10), 5507–5525 (2019).

18. Rasp, S., Pritchard, M. S., Gentine, P.: Deep learning to represent subgrid processes in climate models. Proceedings of the National Academy of Sciences, **115**(39), 9684–9689 (2018).
19. Scher, S.: Toward Data-Driven Weather and Climate Forecasting: Approximating a Simple General Circulation Model with Deep Learning. Geophysical Research Letters, **45**(22), 12–616 (2018).
20. Fraedrich, K., Jansen, H., Kirk, E., Luksch, U., Lunkeit, F.: The Planet Simulator: Towards a user friendly model. Meteorologische Zeitschrift, **14**(3), 299–304 (2005).
21. Dijkstra, H., Hernandez-Garcia, E., Lopez, C.: The application of Machine Learning Techniques to improve El Nino prediction skill. Frontiers in Physics, **7**, 153 (2019).
22. Shearer, P. M.: Introduction to seismology, Third Edition. Cambridge University Press (2019).
23. Havskov, J., Alguacil, G.: Instrumentation in earthquake seismology (Vol. 358). Springer (2004).
24. Schweitzer, J., Fyen, J., Mykkeltveit, S.: Seismic arrays. In IASPEI New Manual of Seismological Observatory Practice, ed. P. Bormann, chapter 9. Potsdam, Germany: GeoForschungsZentrum Potsdam (2002).
25. Bormann, P.: New Manual of Seismological Observatory Practice (NMSOP-2), Chapter 11, IASPEI, GFZ German Research Centre for Geosciences, Potsdam (2012) https://doi.org/10.2312/GFZ.NMSOP-2.
26. Allen, R. V.: Automatic earthquake recognition and timing from single traces. Bulletin of the Seismological Society of America, **68**(5), 1521–1532 (1978).
27. Allen, R.: Automatic phase pickers: Their present use and future prospects. Bulletin of the Seismological Society of America, **72**(6B), S225-S242 (1982).
28. Trnkoczy, A., Bormann, P., Hanka, W., Holcomb, L. G., Nigbor, R. L., Shinohara, M., Shiobara, H., Suyehiro, K.: New Manual of Seismological Observatory Practice 2 (NMSOP-2). Understanding and parameter setting of STA/LTA trigger algorithm, 1–20 (2012).
29. Rabin, N., Bregman, Y., Lindenbaum, O., Ben-Horin, Y., Averbuch, A.: Earthquake-explosion discrimination using diffusion maps. Geophysical Journal International, **207**(3), 1484–1492 (2016).
30. Joswig, M.: Pattern recognition for earthquake detection. Bulletin of the Seismological Society of America, **80** 170–186 (1990).
31. Joswig, M.: Automated processing of seismograms by SparseNet, Seismological Research Letters, **70** 705–711 (1999).
32. Ohrnberger, M.: Continuous automatic classification of seismic signals of volcanic origin at Mt. Merapi, Java, Indonesia. PhD thesis, University of Potsdam (2001).
33. Bregman, Y., Rabin, N.: Aftershock identification using diffusion maps. Seismological Research Letters, **90**(2A), 539–545 (2018).
34. Lindenbaum, O., Rabin, N., Bregman, Y., Averbuch, A.: Multi-channel fusion for seismic event detection and classification. In 2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE) 1–5 (2016).
35. Coifman, R. R., Lafon, S.: Diffusion maps. Applied and Computational Harmonic Analysis, **21**(1), 5–30 (2006).
36. Lindenbaum, O., Bregman, Y., Rabin, N., Averbuch, A.: Multiview kernels for low-dimensional modeling of seismic events. IEEE Transactions on Geoscience and Remote Sensing, **56**(6), 3300–3310 (2018).
37. Sharma, B. K., Kumar, A., Murthy, V. M.: Evaluation of seismic events detection algorithms. Journal of the Geological Society of India, **75**(3), 533–538 (2010).
38. Saragiotis, C. D., Hadjileontiadis, L. J., Panas, S. M.: PAI-S/K: A robust automatic seismic P phase arrival identification scheme. IEEE Transactions on Geoscience and Remote Sensing, **40**(6), 1395–1404 (2002).
39. Galiana-Merino, J. J., Rosa-Herranz, J. L., Parolai, S.: Seismic P Phase Picking Using a Kurtosis-Based Criterion in the Stationary Wavelet Domain. IEEE Transactions on Geoscience and Remote Sensing, **46**(11), 3815–3826 (2008).

40. Küperkoch, L., Meier, T., Lee, J., Friederich, W., & EGELADOS Working Group.: Automated determination of P-phase arrival times at regional and local distances using higher order statistics. Geophysical Journal International, **181**(2), 1159–1170 (2010).
41. Taylor, K. M., Procopio, M. J., Young, C. J., Meyer, F. G.: Estimation of arrival times from seismic waves: a manifold-based approach. Geophysical Journal International, **185**(1), 435–452 (2011).
42. Ramirez Jr, J., Meyer, F. G.: Machine learning for seismic signal processing: Phase classification on a manifold. In 2011 10th International Conference on Machine Learning and Applications and Workshops, **1**, 382–388 (2011).
43. Leonard, G., Villagran, M., Joswig, M., Bartal, Y., Rabinowitz, N., Saya, A.: Seismic source classification in Israel by signal imaging and rule-based coincidence evaluation. Bulletin of the Seismological Society of America, **89**(4), 960–969 (1999).
44. Parolai, S., Trojani, L., Frapiccini, M., Monachesi, G.: Seismic source classification by means of a sonogram-correlation approach: application to data of the RSM seismic network (Central Italy). Pure and Applied Geophysics, **159**(11–12), 2763–2788 (2002).
45. Sick, B., Walter, M., Joswig, M.: Visual event screening of continuous seismic data by supersonograms. Pure and Applied Geophysics, **171**(3–5), 549–559 (2014).
46. Vouillamoz, N., Wust-Bloch, G. H., Abednego, M., Mosar, J.: Optimizing Event Detection and Location in Low-Seismicity Zones: Case Study from Western Switzerland. Bulletin of the Seismological Society of America, **106**(5), 2023–2036 (2016).
47. Ishida, M., Kanamori, H.: The foreshock activity of the 1971 San Fernando earthquake, California. Bulletin of the Seismological Society of America, **68**(5), 1265–1279 (1978).
48. Geller, R. J., Mueller, C. S.: Four similar earthquakes in central California. Geophysical Research Letters, **7**(10), 821–824 (1980).
49. Harris, D. B.: A waveform correlation method for identifying quarry explosions. Bulletin of the Seismological Society of America, **81**(6) 2395–2418 (1991).
50. Withers, M., Aster, R., Young, C.: An automated local and regional seismic event detection and location system using waveform correlation, Bulletin of the Seismological Society of America, **89**(3) 657–669 (1999).
51. Gibbons, S. J., Ringdal, F.: The detection of low magnitude seismic events using array-based waveform correlation. Geophysical Journal International, **165**, 149–166 (2006).
52. Schaff, D. P., Waldhauser, F.: Improving magnitude detection thresholds using multi-event, multi-station, and multi-phase methods, SSA Annual Meeting 18–22 April 2006.
53. Slinkard, M. E., Carr, D. B., Young, C. J.: Applying waveform correlation to three aftershock sequences. Bulletin of the Seismological Society of America, **103**(2A) 675–693 (2013).
54. Bobrov, D., Kitov, I., Zerbo, L.: Perspectives of cross-correlation in seismic monitoring at the International Data Centre. Pure and Applied Geophysics, **171**(3/5) 439–468 (2014).
55. Ganter, T., Sundermier, A., Ballard, S.: Alternate Null Hypothesis Correlation: A New Approach to Automatic Seismic Event Detection. Bulletin of the Seismological Society of America, **108**(6), 3528–3547 (2018).
56. Yoon, C. E., O'Reilly, O., Bergen, K. J., Beroza, G. C.: Earthquake detection through computationally efficient similarity search. Science Advances, **1**(11), e1501057 (2015).
57. Bergen, K. J., Beroza, G. C.: Detecting earthquakes over a seismic network using single-station similarity measures. Geophysical Journal International, **213**(3), 1984–1998 (2018).
58. Bergen, K. J., Beroza, G. C.: Earthquake fingerprints: Extracting waveform features for similarity-based earthquake detection. Pure and Applied Geophysics, **176**(3), 1037–1059 (2019).
59. Yoon, C. E., Bergen, K. J., Rong, K., Elezabi, H., Ellsworth, W. L., Beroza, G. C., Bailis, P., Levis, P.: Unsupervised Large-Scale Search for Similar Earthquake Signals. Bulletin of the Seismological Society of America (2019).
60. Baluja, S., Covell, M.: Waveprint: Efficient wavelet-based audio fingerprinting. Pattern Recognition, **41**(11), 3467–3480 (2008).
61. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), 459–468 (2006).

62. Rong, K., Yoon, C. E., Bergen, K. J., Elezabi, H., Bailis, P., Levis, P., Beroza, G. C.: Locality-sensitive hashing for earthquake detection: A case study of scaling data-driven science. Proceedings of the VLDB Endowment, **11**(11), 1674–1687 (2018).
63. Blandford, R. R.: Seismic event discrimination. Bulletin of the Seismological Society of America, **72**(6B), S69-S87 (1982).
64. Rodgers, A. J., Lay, T., Walter, W. R., Mayeda, K. M.: A comparison of regional-phase amplitude ratio measurement techniques. Bulletin of the Seismological Society of America, **87**(6), 1613–1621 (1997).
65. Kortström, J., Uski, M., Tiira, T: Automatic classification of seismic events within a regional seismograph network. Computers & Geosciences, **87**, 22–30 (2016).
66. Hafemeister, D.: Progress in CTBT monitoring since its 1999 Senate defeat. Science & Global Security, **15**(3), 151–183 (2007).
67. Arora, N. S., Russell, S., Sudderth, E.: NET-VISA: Network processing vertically integrated seismic analysis. Bulletin of the Seismological Society of America, **103**(2A), 709–729 (2013).
68. Arora, N. S., Russell, S., Kidwell, P., Sudderth, E.: Global seismic monitoring: A Bayesian approach. In Twenty-Fifth AAAI Conference on Artificial Intelligence (2011).
69. Tiira, T.: Discrimination of nuclear explosions and earthquakes from teleseismic distances with a local network of short period seismic stations using artificial neural networks. Physics of the Earth and Planetary Interiors, **97** 1–4, 247–268 (1996).
70. AllamehZadeh, M.: Discrimination analysis of earthquakes and man-made events using ARMA coefficients determination by artificial neural networks. Natural Resources Research, **20**(4), 367–375 (2011).
71. Del Pezzo, E., Esposito, A., Giudicepietro, F., Marinaro, M., Martini, M., Scarpetta, S.: Discrimination of earthquakes and underwater explosions using neural networks. Bulletin of the Seismological Society of America, **93**(1), 215–223 (2003).
72. Esposito, A. M., Giudicepietro, F., Scarpetta, S., D'Auria, L., Marinaro, M., Martini, M.: Automatic discrimination among landslide, explosion-quake, and microtremor seismic signals at Stromboli volcano using neural networks. Bulletin of the Seismological Society of America, **96**(4A), 1230–1240 (2006).
73. Kislov, K. V., Gravirov, V. V.: (2017). Use of artificial neural networks for classification of noisy seismic signals. Seismic Instruments, **53**(1), 87–101 (2017).
74. Hannibal, A. E.: On the Possibility of Using Artificial Neural Networks in Seismic Monitoring Tasks. Seismic Instruments, **55**(3), 334–344 (2019).
75. Mousavi, S. M., Horton, S. P., Langston, C. A., Samei, B.: Seismic features and automatic discrimination of deep and shallow induced-microearthquakes using neural network and logistic regression. Geophysical Journal International, **207**(1), 29–46 (2016).
76. Giudicepietro, F., Esposito, A. M., Ricciolino, P.: Fast discrimination of local earthquakes using a neural approach. Seismological Research Letters, **88**(4), 1089–1096 (2017).
77. Kuyuk, H. S., Yildirim, E., Dogan, E., Horasan, G.: An unsupervised learning algorithm: application to the discrimination of seismic events and quarry blasts in the vicinity of Istanbul. Natural Hazards and Earth System Sciences, **11**(1), 93–100 (2011).
78. Kohonen, T.: Self Organization Maps. Springer Series in Information Sciences, vol. 30. Third Extended Edition, 501 pp, Springer Berlin, Heidelberg, New York, 1995, 1997, (2001).
79. Köhler, A., Ohrnberger, M., Scherbaum, F.: Unsupervised pattern recognition in continuous seismic wavefield records using self-organizing maps. Geophysical Journal International, **182**(3), 1619–1630 (2010).
80. Sick, B., Guggenmos, M., Joswig, M.: Chances and limits of single-station seismic event clustering by unsupervised pattern recognition. Geophysical Journal International, **201**(3), 1801–1813 (2015).
81. Abdi, H., Williams, L. J.: Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, **2**(4), 433–459 (2010).

82. Reynen, A., Audet, P.: Supervised machine learning on a network scale: Application to seismic event classification and detection. Geophysical Journal International, **210**(3), 1394–1409 (2017).
83. Vapnik, V. N.: The Nature of Statistical Learning Theory. Springer, Berlin (1995).
84. Saad, O. M., Shalaby, A., Inoue, K., Sayed, M. S.: Hardware Friendly Algorithm for Earthquakes Discrimination Based on Wavelet Filter Bank and Support Vector Machine. In 2018 International Japan-Africa Conference on Electronics, Communications and Computations (JAC-ECC) 115–118 (2018).
85. Saad, O. M., Shalaby, A., Sayed, M. S.: Automatic discrimination of earthquakes and quarry blasts using wavelet filter bank and support vector machine. Journal of Seismology, **23**(2), 357–371 (2019).
86. Hickmann, K. S., Hyman, J., Srinivasan, G.: Efficient and robust classification of seismic data using nonlinear support vector machines. In 2017 51st Asilomar Conference on Signals, Systems, and Computers 148–155 (2017).
87. Beyreuther, M., Wassermann, J.: Continuous earthquake detection and classification using discrete Hidden Markov Models. Geophysical Journal International, 175 1055–1066 (2008).
88. Beyreuther, M., Carniel, R., Wassermann, J.: Continuous hidden Markov models: application to automatic earthquake detection and classification at Las Canãdas caldera, Tenerife. Journal of Volcanology and Geothermal Research, **176**(4), 513–518 (2008).
89. Bicego, M., Acosta-Muñoz, C., Orozco-Alzate, M.: Classification of seismic volcanic signals using hidden-Markov-model-based generative embeddings. IEEE Transactions on Geoscience and Remote Sensing, **51**(6), 3400–3409 (2012).
90. Hammer, C., Beyreuther, M., Ohrnberger, M.: A seismic-event spotting system for volcano fast-response systems. Bulletin of the Seismological Society of America, **102**(3), 948–960 (2012).
91. Beyreuther, M., Hammer, C., Wassermann, J., Ohrnberger, M., Megies, T.: Constructing a Hidden Markov Model based earthquake detector: application to induced seismicity. Geophysical Journal International, **189**(1), 602–610 (2012).
92. Quang, P. B., Gaillard, P., Cano, Y., Ulzibat, M.: Detection and classification of seismic events with progressive multi-channel correlation and hidden Markov models. Computers & Geosciences, **83**, 110–119 (2015).
93. Hammer, C., Ohrnberger, M., Faeh, D.: Classifying seismic waveforms from scratch: a case study in the Alpine environment. Geophysical Journal International, **192**(1), 425–439 (2012).
94. Riggelsen, C., Ohrnberger, M., Scherbaum, F.: Dynamic Bayesian networks for real-time classification of seismic signals. In European Conference on Principles of Data Mining and Knowledge Discovery, 565–572, Springer (2007).
95. Riggelsen, C., Ohrnberger, M.: A machine learning approach for improving the detection capabilities at 3C seismic stations. Pure and Applied Geophysics, **171**(3–5), 395–411 (2014).
96. Kuyuk, H. S., Yildirim, E., Dogan, E., Horasan, G.: Application of k-means and Gaussian mixture model for classification of seismic activities in Istanbul. Nonlinear Processes in Geophysics, **19**(4), 411–419 (2012).
97. Kong, Q., Allen, R. M., Schreier, L., Kwon, Y. W.: MyShake: A smartphone seismic network for earthquake early warning and beyond. Science Advances, **2**(2), e1501055 (2016).
98. Kong, Q., Inbal, A., Allen, R. M., Lv, Q., Puder, A.: Machine learning aspects of the MyShake global smartphone seismic network. Seismological Research Letters, **90**(2A), 546–552 (2019).
99. Perol, T., Gharbi, M., Denolle, M.: Convolutional neural network for earthquake detection and location. Science Advances, **4**(2), e1700578 (2018).
100. Lomax, A., Michelini, A., Jozinović, D.: An investigation of rapid earthquake characterization using single-station waveforms and a convolutional neural network. Seismological Research Letters, **90**(2A), 517–529 (2019).
101. Li, Z., Meier, M. A., Hauksson, E., Zhan, Z., Andrews, J.: Machine learning seismic wave discrimination: Application to earthquake early warning. Geophysical Research Letters, **45**(10), 4773–4779 (2018).
102. Kuyuk, H. S., Susumu, O.: Real-Time Classification of Earthquake using Deep Learning. Procedia Computer Science, **140**, 298–305 (2018).

103. Zhu, W., Mousavi, S. M., Beroza, G. C.: Seismic signal denoising and decomposition using deep neural networks. IEEE Transactions on Geoscience and Remote Sensing, **57**(11), 9476–9488 (2019).
104. Ross, Z. E., Meier, M. A., Hauksson, E.: P wave arrival picking and first-motion polarity determination with deep learning. Journal of Geophysical Research: Solid Earth, **123**(6), 5120–5129 (2018).
105. Zhu, W., Beroza, G. C.: PhaseNet: a deep-neural-network-based seismic arrival-time picking method. Geophysical Journal International, **216**(1), 261–273 (2018).
106. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention, 234–241, Springer (2015).
107. Woollam, J., Rietbrock, A., Bueno, A., De Angelis, S.: Convolutional neural network for seismic phase classification, performance demonstration over a local seismic network. Seismological Research Letters, **90**(2A), 491–502 (2019).
108. Rietbrock, A., Ryder, I., Hayes, G., Haberland, C., Comte, D., Roecker, S., Lyon-Caen, H.: Aftershock seismicity of the 2010 Maule Mw= 8.8, Chile, earthquake: Correlation between co-seismic slip models and aftershock distribution?. Geophysical Research Letters, **39**(8) (2012).
109. Ross, Z. E., Yue, Y., Meier, M. A., Hauksson, E., Heaton, T. H.: PhaseLink: A deep learning approach to seismic phase association. Journal of Geophysical Research: Solid Earth, 124(1), 856–869 (2019).
110. McBrearty, I. W., Delorey, A. A., Johnson, P. A.: Pairwise association of seismic arrivals with convolutional neural networks. Seismological Research Letters, **90**(2A), 503–509 (2019).
111. Kriegerowski, M., Petersen, G. M., Vasyura-Bathke, H., Ohrnberger, M.: A deep convolutional neural network for localization of clustered earthquakes based on multistation full waveforms. Seismological Research Letters, **90**(2A), 510–516 (2018).
112. Nakano, M., Sugiyama, D., Hori, T., Kuwatani, T., Tsuboi, S.: Discrimination of seismic signals from earthquakes and tectonic tremor by applying a convolutional neural network to running spectral images. Seismological Research Letters, **90**(2A), 530–538 (2019).
113. Linville, L., Pankow, K., Draelos, T.: Deep Learning Models Augment Analyst Decisions for Event Discrimination. Geophysical Research Letters, **46**(7), 3643–3651 (2019).
114. Tibi, R., Linville, L., Young, C., Brogan, R.: Classification of Local Seismic Events in the Utah Region: A Comparison of Amplitude Ratio Methods with a Spectrogram-Based Machine Learning Approach. Bulletin of the Seismological Society of America, **109**(6), 2532–2544 (2019).
115. Lindenbaum, O., Rabin, N., Bregman, Y., Averbuch, A.: Seismic Event Discrimination Using Deep CCA. IEEE Geoscience and Remote Sensing Letters, to appear (2019).
116. Andrew, G., Arora, R., Bilmes, J., Livescu, K.: Deep canonical correlation analysis. In International Conference on Machine Learning, 1247–1255 (2013).
117. Zhu, L., Liu, E., McClellan, J. H.: Sparse-promoting full-waveform inversion based on online orthonormal dictionary learning. Geophysics, **82**(2), R87–R107 (2017).
118. Li, D., Harris, J. M.: Full waveform inversion with nonlocal similarity and model-derivative domain adaptive sparsity-promoting regularization. Geophysical Journal International, **215**(3), 1841–1864 (2018).
119. Moseley, B., Nissen-Meyer, T., Markham, A.: Deep learning for fast simulation of seismic waves in complex media. Solid Earth, **11**(4), 1527–1549 (2020).
120. Li, S., Liu, B., Ren, Y., Chen, Y., Yang, S., Wang, Y., Jiang, P.: Deep learning inversion of seismic data. IEEE Transactions on Geoscience and Remote Sensing, **58**(3), 2135–2149 (2020).
121. Alavi, A. H., Gandomi, A. H.: Prediction of principal ground-motion parameters using a hybrid method coupling artificial neural networks and simulated annealing. Computers & Structures, **89**(23–24), 2176–2194 (2011).
122. Derras, B., Bard, P-Y., Cotton, F., Bekkouche, A.: Adapting the neural network approach to PGA prediction: An example based on the KiK-net data. Bulletin of the Seismological Society of America, **102**(4), 1446–1461 (2012).

123. Trugman, D. T., Shearer, P. M.: Strong correlation between stress drop and peak ground acceleration for recent M 1–4 earthquakes in the San Francisco Bay area. Bulletin of the Seismological Society of America **108**(2), 929–945 (2018).
124. Rouet-Leduc, B., Hulbert, C., Lubbers, N., Barros, K., Humphreys, C. J., Johnson, P. A.: Machine learning predicts laboratory earthquakes. Geophysical Research Letters **44**(18), 9276–9282 (2017).
125. Rouet-Leduc, B., Hulbert, C., Bolton, D. C., Ren, X. C., Riviere, J., Marone, C., Guyer, A. R., Johnson, A. P.: Estimating fault friction from seismic signals in the laboratory. Geophysical Research Letters, **45**(3), 1321–1329 (2018).
126. Los Alamos National Laboratory (2019). LANL earthquake prediction, Kaggle, available at https://www.kaggle.com/c/LANL-Earthquake-Prediction/overview (last assessed 27 September 2020).
127. Mignan, A., Broccardo, M.: Neural Network Applications in Earthquake Prediction (1994–2019): Meta-Analytic and Statistical Insights on Their Limitations. Seismological Research Letters, **90**(4), 2330–2342 (2020).

Sentiment Analysis for Social Text

Nir Ofek

1 Introduction

This work is an essential map for sentiment analysis (SA) in social media. Instead of providing the readers with a pull of methods, we chose to provide a comprehensive map to understand the field. Methods that were explicated more thoroughly in each section are brought due to several reasons: they are robust to noisy text and therefore particularly suitable to be used with social text, and they allow easy fusion with other methods.

Sentiment analysis, which is also called *opinion mining*, is the computational detection and study of opinions and viewpoints underlying a text span; it enables organizations to sound out public or consumer opinions and paves the way for researchers to better model and understand a large variety of human communications in order to provide significant insight into the sentiments, emotions, thoughts, and opinions of people. Examples include analysis of social support [1], distilling personality from language [2], detecting political opinions [3], and even measuring the quality of life [4]; gauging the quality of various products in reviews and microblogs has become the de-facto standard for assessing the quality of products and services [5]. The unprecedented popularity and evolution of social communication platforms has led to an abundance of user-generated content (UGC) that has become the primary source of information for sentiment analysis.

The focus of this chapter is sentiment analysis in social text, which is typically short, informal, and is generated by users in a variety of social media platforms, such as microblogs, forums, and social networks. We focus on the English language, arguably the most widely used language in the world, especially as a lingua franca;

N. Ofek (✉)
Ben-Gurion University of the Negev Beer Sheva, Be'er Sheva, Israel
e-mail: nirofek@post.bgu.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_35

however, most of the presented methods can be easily adopted to other languages. Our choice to focus on platforms that allow and encourage personal expressions, e.g., Twitter, online health communities, and Websites that support product and service rating (e.g., TripAdvisor and Amazon), stems from the understanding that they are integral part of human communications. Apart from text, social data may include various of modalities such as image and voice; for example, users' profile information can be utilized to rate their level of expertise in the discussed topic and rate their opinions accordingly [6]. In this chapter, we explore methods for textual data itself, overlooking any other contextual data, since it is the leading and principal resource for sentiment discovery and can be fused with non-textual and contextual information.

Making meaning out of unstructured information is extremely difficult since text, despite being perfectly suitable for human consumption, is largely unintelligible for machines. Challenges include versatility, namely, the multiple variations to convey the same meaning, and the tendency to not comply with simple rules, particularly in UGC, where characters are limited, and often text is grammatically erroneous, without punctuation marks and informal. Compared with other natural language processing (NLP) challenges, such as simple text categorization, sentiment analysis relies heavily on understanding of the context, e.g., negation. In addition, some words do not carry any specific polarity of their own but acquire it in context: for example, the adjective *small* is usually positive in the context of *problem* and negative in the context of *hotel room*.

There are various of approaches to address sentiment analysis; the granularity level of sentiment and the problem setting are the main factors to consider in choosing an approach. The level of granularity defines the sentiment resolution. This is important when we want to capture multiple opinions within the same sentence, since computing sentiment at the sentence level will not consider divergent opinions. Consider the following review: "the treatment was successful but the staff were unpleasant." A positive viewpoint is expressed regarding the *treatment*, while a negative opinion is expressed toward the *staff*. The brevity of social text (e.g., the character limitation of Twitter) invites for fine analysis of sentiment. This also allows in-depth analysis, such as investigating sentiment change in an online community (e.g., change within threads, by detecting opinions at the post-level within forum threads), without having to directly survey the population, a time-consuming and expensive task [7].

The characteristics of the available data determine the approach we will adopt or avoid. For example, we may avoid using parsers for noisy text that hardly adhere to grammatical rules since this causes a decline in the performance of NLP techniques [8]. Other characteristics include the availability of annotated data and its quantity. Due to the above-mentioned, there is no single robust method for sentiment analysis. The level of granularity depends on the desired outcome, e.g., do we want to be sensitive to divergent opinions within the same sentence? The problem setting invites different approaches, e.g., supervised or unsupervised, depending on the availability of annotated training data. Supervised learning is less relevant for aspect-level SA because it is impractical to obtain annotations for every opinion

target (aspect). This chapter guides through the relevant approaches for various of problem settings.

Current trends in sentiment analysis involve deep learning (DL) approaches that had been proven to function well in various of NLP tasks [9]. The main advantage of employing deep learning approach lies in its independence from expert knowledge, since the text representation is learnt according to the task. However, many methods are not specific for sentiment analysis, and their merit to the understanding of the problem is limited, despite the fact that they can work well. See further discussion in Sect. 7.

This chapter is organized by first mapping the field and its relevant terminology. It proceeds with three main tasks for SA in social data, according to their granularity. In lexical level, the task is to create a set of single expressions along with a measure of their polarity. In aspect-level SA, the task is to capture opinions per opinion targets; in this section, there is a special sensitivity to the availability of annotated data because it is impractical to obtain sufficient labeled data for every target aspect (e.g., for screen, price, weight, and battery life in the context of smartphone). Finally, in sentence-level SA, we regard a sentence as any fragment of text. Each task can support additional tasks. For example, document-level SA (which is less prevalent in social data) can be supported by sentence-level analysis by considering document as a collection of (ordinal) sentences. At the end of the chapter, we describe several applications and continue to discuss the sentiment problem and to raise questions about ethics in SA.

1.1 Ontology

In this section, we provide a map of concepts related to sentiment analysis on user-generated text.

Opinion The field of sentiment analysis is mainly referred to as the task of identifying the polarity, namely positive or negative, of a text fragment, or identifying the level of being positive. It seeks to develop automatic methods to address the problem. In the literature, opinion mining and sentiment analysis primarily refer to the same computational task that is generally ascribed to the detection of the viewpoints underlying a text span [10]. The task refers to as opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, and review mining—all fall under the umbrella of opinion mining.

Sentiment Word Sentiment word refers to a word that expresses opinion by itself, although its polarity may depend on the context. Sentiment words are instrumental for many sentiment analysis applications and include words with strong sentiment such as “terrible” and “exiting” or words with weaker sentiment such as “considerable” and “nice.”

Sentiment Level Sentiment can be computed in many levels of granularity: at a document or sentence level, for specific target word and for a single word (or phrase) in inself. However, detecting opinions at the document level is not relevant in this work due to the brevity of UGC. Sentence-level SA is often insufficient for applications because it does not assign sentiments to opinion targets [11]. Aspect-level SA aims to associate sentiment words with their opinion targets, i.e., aspects. In word/phrase-level SA, a sentiment lexicon is created. Choosing the sentiment level depends on the task, and there is a link between the various levels. For example, finding polarity of aspects and expressions can support sentiment analysis at various granularity levels, since the orientation of a text fragment can be determined by the polarity of its known components. The organization of this chapter follows a bottom-up approach. The more fine-grained approaches are discussed first.

Sentiment Lexicon Sentiment lexicon is a knowledge base of words or phrases along with their sentiment score. It is a fundamental component for sentiment analysis and can support sentiment analysis in all levels of granularity. Lexical words such as "amazing," "horrible," and "excellent," are assigned with a positive or negative score reflecting their sentiment polarity and strength.

Sentiment Score The sentiment score is the output of a sentiment computation task. In some cases, sentiment analysis aims to predict the rating of a review, as in aspect-level SA where rating (from one to five) is often predicted; in others, the goal is to discriminate between binary classes (positive or negative), or in a trinary scale (positive, neutral, and negative categories). This work presents methods that are relevant to the common scheme prediction.

Ambiguity Unlike simple text categorization, opinion mining relies heavily on understanding of the context, since some words do not carry any specific polarity of their own or are ambiguous. For example, the adjective "great" is likely to be positive in the context of an *idea* and negative in the context of a *problem*. To address this, a popular approach aims to couple sentiment words along with their contextual words to disambiguate their polarity.

Classification vs. Extraction Sentiment is computed in two main approaches: the extraction approach and the classification approach. The extraction approach aims to identify words, multiword expressions, phrases, or clauses that express attitudes and determine the polarity of these attitudes [12]. Therefore, it often involves a sentiment lexicon where words or phrases are coined with their polarity orientation. This allows calculation of sentiment in any fragment of text by extracting its known components from the lexicon and building the sentiment picture accordingly, as shown by Turney [13]. The classification approach involves building classifiers from labeled texts (essentially a supervised classification task).

Supervised vs. Unsupervised Typically, SA is handled by supervised classification, and the goal is to discriminate between the two classes. Supervised machine learning methods rely on the availability of quality labelled data and are desired because of their ability to focus on the specific resources of interest. In SA, often

positive examples outnumber the negative examples; in this case, the imbalance class distribution problem may occur, because there are significantly more instances from one class relative to other classes [14]. In such cases, the classifier tends to misclassify the instances of the less represented classes, for a variety of reasons. There are several approaches to address this problem, as discussed and suggested by Ofek et al. [15]. The unsupervised approach tends to be less labor-intensive, and the extraction approach becomes more relevant by utilizing rules and lexicons. For example, bootstrapping techniques can be employed to label data according to syntactic patterns, and clustering techniques can form clusters according to the feature space, while labels can be manually or heuristically obtained. In this scope, there are also methods that utilize signals such as emoticons, to obtain labeled data, or utilize given label information in their optimization function, while the predicted target class is different; see, for example, the weakly supervised approach in Sect. 4.

2 Text Representation and Normalization

Text, an unstructured type of data, requires representation that bridges the human understanding of language to that of a machine. This process is conducted by an expert or by machines that learn representation. Below are several concerns related to text mining in the context of sentiment analysis.

Bag-of-Words (BOW) Word models represent text as a fixed-length vector; if each term occurring in the training documents is used as a feature, the vector length equals to the distinct n-grams in the vocabulary. Term frequency (TF) vectors [16] are used to represent terms as their normalized counts; the distance between vectors can be measured by employing cosine similarity or other measures. Although it is not without merit, the BOW approach is often incapable of representing the order and proximity of terms in the text. This poses significant limitations in some sentiment analysis scenarios, as explained in [17]. Another major drawback of BOW-based solutions is the difficulty of integrating additional information into the model because of the large number of features it utilizes. This problem is known as the curse of dimensionality.

Curse of Dimensionality Many text classification methods use sparse features representation, mainly terms and their frequencies, in which each feature is the frequency of an n-gram unit. Despite being simple, this approach is highly effective for traditional text classification, and in particular for some sentiment analysis tasks [18]. Since this approach involves high feature space, classifiers find it difficult to utilize them in an efficient manner. This problem is known as the curse of dimensionality [19, 20]. The large number of features also narrows the selection of possible classifiers, ruling out popular algorithms such as the C4.5 decision trees algorithm [21], which are incapable of efficiently dealing with a feature set of large magnitude. This problem is considerably aggravated when the training dataset is relatively small and can lead to overfitting of the learned models. Feature abstraction

methods have been shown to effectively reduce the number of parameters without sacrificing classification accuracy as, demonstrated by [22, 23], and by conSent method in Sect. 5.

Word Embedding Word embedding learns semantically meaningful representations for words and encodes them in fixed-length vectors, so similar words have similar representation. Often the principle is to describe words by its frequent co-occurring words in a large text corpus, using unsupervised methods. In practice, word embedding techniques such as word2vec [24] learn to predict a word given the other words in its vicinity. In this space, vectors embody semantic relatedness of words, such that "bee" is closer to "butterfly," than to "Japan," which enables the effective comparison between words. This representation can be used to feed any feature-based technique.

Text Embedding Sentence or document embedding models [25] map each fragment of text to a dense, low-dimensional vector, in continuous vector space. Sentence embedding vectors take into consideration the word order while inheriting an important property of the word vectors: the semantics of the words, so semantically related texts, should be represented by close vectors. In sentiment analysis, a sentence can be represented by such manner to feed any classifier, while the choice of text representation is often more important than the type of classifier used.

Negation Detection Negation is a very common linguistic construction that affects the semantic orientation of words, and therefore, negation detection is important to sentiment analysis. In order to detect negation, a simple technique can be adopted. The following prefixes are considered as negating: mis-, un-, dis-, and im-. Additionally, negation words can be adopted from [26], and consider a word as negated if it is dependent on a negated word. In case a parser is not available to draw the dependency tree, other methods can be implemented as described by [8, 27]. Further discussion and analysis of negation detection are beyond the scope of this work and can be reviewed in [28], and negation words can be found at Christopher Potts' tutorial. http://sentiment.christopherpotts.net/lingstruc.html.

Normalization Textual irregularities in UGC are numerous and diverse; normalizing them may result in reducing the level of noise and the dimensionality of the data, e.g., by bringing multiple variations that convey the same meaning, to a single normal form. Below are several irregularities in Twitter, whose grammatical structure is different from common structure of longer texts (e.g., new articles). The limitation on length of tweets (140 characters) encourages the use of abbreviations; the letter "c," which is included among the top 100 most frequently used terms according to Google's Twitter frequency lexicon, is often used instead of the word "see." Other irregularities include a lack of punctuation marks, the use of informal text, slang, non-standard shortcuts, and words concatenations. For example, in Twitter, the hashtag #coronavirusoutbreak could be converted into "corona virus outbreak," and user names can be mapped into a single token.

2.1 Resources

NLTK. The Natural Language Toolkit (NLTK) is a Python platform for performing NLP-related tasks, e.g., tokenization, Part-Of-Speech (POS) tagging, stemming, parsing, and semantic reasoning. NLTK also provides interfaces for many corpora and lexicons that are useful for and sentiment analysis. http://www.nltk.org/

CoreNLP Stanford CoreNLP is a Java framework that supports NLP task, e.g., named entity recognization, parsing, POS tagging, coreference resolution, and some sentiment analysis capabilities. http://stanfordnlp.github.io/CoreNLP/.

Gensim Gensim is a Python open-source library for topic modeling with large-scale capabilities; it suggests word2vec implementation, online latent semantic analysis (LSA), latent Dirichlet allocation (LDA), random projection, and hierarchical Dirichlet process. http://radimrehurek.com/gensim/.

3 Lexical-Level Sentiment Analysis

Words are fundamental component tokens in human language; knowing words' sentiment is a basic step in sentiment analysis, because it can be utilized to compute sentiment of any fragment of text, under the assumption that individual words have what is referred to as prior polarity. Prior polarity is referred to as the sentiment orientation independent of context. In this section, we introduce methods to construct a sentiment lexicon that comprised of sentiment words along with a measure of their polarity, to indicate the direction the word deviates from the norm of its semantic group. Apart from words, lexical components may include expressions and concepts.

The basic approach for lexicon construction assembles manually a small list of seed words, along with their measure of polarity. Then, an algorithm is adopted to propagate sentiment information to words that relate to the seed words. In more details, first seed words (word or phrase) are added as nodes. Second, a graph is expanded by adding related words to each seed word as new nodes, while the similarity with known words is considered as weighted edges. Then, graph propagation algorithms, such as PageRank [29], label propagation [30], implemented in [31], or random walk (used in constructing SentiWordNet [32]), are utilized to iteratively compute the sentiment score of new words. Two main approaches are taken in selecting the data source for the graph expansion process. One is corpus-based, where rules, linguistic patterns, or simple words co-occurrences aim to define the sentiment relationship between pairs of words. In this case, sentiment of new words is computed based on a large number or co-occurrence with known words. Obtaining a large corpus will not only improve the accuracy of words' sentiment, but also increase the lexicon coverage; the challenge here is to maintain accuracy

by, e.g., using a rigid set of rules or patterns, in order to ensure a quality inference process while obtaining a reasonable coverage.

Turney and Littman [33] conducted a seminal study in this line of work; they count word co-occurrences by feeding queries to a search engine and collecting the number of hits; therefore, the computation is based on a very large dataset, namely, the entire indexed Web. Then, they employ pointwise mutual information (PMI) between two items that measures the degree of statistical dependence between two terms, such that:

$$PMI(word_1, word_2) = log_2 \frac{p(word_1 \& word_2)}{p(word_1)p(word_2)},$$

where $p(word_1 \& word_2)$ is the probability that $word_1$ and $word_2$ co-occur. If the words are statistically independent, the probability that they co-occur is given by the product $p(word_1)p(word_2)$. The ratio given by $p(word_1 \& word_2)$ and $p(word_1)p(word_2)$ is a measure of the degree of statistical dependence between the words. The log of the ratio corresponds to positive correlation when the words tend to co-occur and negative when they do not.

The second approach for lexicon construction is thesaurus-based. It exploits semantic relationships (e.g., synonym and antonym) in a dictionary to expand the words graph. As a consequence, under this direction, the majority of existing works regard word as their basic unit [34].

SentiWordNet [32] is a lexical resource for sentiment analysis that assigns to each synset of WordNet [35]—a lexical database for English—three sentiment scores: positive, negative, and objective. Since it is indexing the meaning of a word, a word can be associated with several sentiment scores, according to its meaning. The lexicon is generated by employing expansion of words from small sets of seed words. Here are the main steps of the process:

1. First, seven positive and seven negative terms are collected and considered as seed set.
2. In this step, the seed set is automatically expanded. The process involves traversing WordNet in several iterations to enrich the lexicon by words of similar polarity (by, e.g., a relation of synsets) and of opposite polarity (a relation of antonymy).
3. Here, a third collection of objective words is added. They assume that terms with a similar polarity tend to have "similar" glosses—a brief textual definition of each synset in WordNet, used for sense disambiguation. Therefore, synset glosses are labeled with one of three sentiment classes: positive, negative, and objective. The glosses are used for training so as the ternary model is actually a gloss classifier (in this case averaging ensemble of classifiers).
4. The classifier trained in the previous step is used to classify all synsets in WordNet.

Word Sentiment Embedding Word sentiment embedding is another approach for lexicon construction where words are represented in continuous vector space;

however, words that are likely to appear in similar contexts, from a sentiment point of view, should not necessarily have similar representations in typical embeddings. For example, the words "safe" and "unsafe" can appear in similar contexts. By merely looking at word co-occurrences, we would learn similar vector representations for "safe" and "unsafe." To capture their opposite polarity, there is a need to incorporate sentiment knowledge in the learning process so as better distributional representations for SA could be learnt. The study of [36] suggests considering sentiment information in word embedding, by incorporating a measure of sentiment similarity so as to capture the sentiments from prior information; they compound sentiment similarity as a ratio for appearing in every sentiment category, with co-occurrence measure, so as similar words by context, such as "safe" and "unsafe," which have different sentiment distributions, correspond to different vector representations.

SSPE The sentiment-specific phrase embedding [37] generates Twitter-specific sentiment lexicon of words and phrases. The embedding phase starts with collecting tweets containing positive and negative emoticons and considering all phrases in a positive tweet as positive and similarly for negative tweets. Next, they employ learning phrase representation with Skip-Gram model for phrase embedding [38]. However, in addition to that, their objective includes predicting the polarity of the whole sentence representation, given by averaging the embedding of its phrases.

The main limitation of the lexical approaches is the ambiguity of words. This can be alleviated by adding more context to words themselves. Thus, it is suggested to construct text into small meaning units, i.e., concepts—semantic forms of human language—and assigning emotions to such concepts.

SenticNet SenticNet [39] is a popular knowledge base of concepts (compiled based on commonsense), along with their polarity scores. Using polarity of natural language concepts can mitigate the complexity of sentiment analysis, since understanding concepts does not require a great deal of familiarity with the language. Concepts may consist of a product's feature described by an opinion word (small room) or an expression (keep alive). Concept-level sentiment analysis aims to infer the semantics associated with natural language opinions and thereby facilitating comparative fine-grained feature-based sentiment analysis [40]. Such knowledgebase enables a deeper and multifaceted analysis of natural language opinions; since it does not contain domain-specific knowledge, the majority of these concept's sentiments are unambiguous.

SentiLARE SentiLARE [41] acquires the linguistic knowledge of each word, to compute its sentiment polarity via a context-aware sentiment attention mechanism over all the matched senses in SentiWordNet. Using the lexical information, they achieve state-of-the-art performance on a several aspect- and sentence-level sentiment analysis tasks.

3.1 Resources

SentiWordNet [32] is a lexical resource for sentiment analysis that assigns polarity to each synset of WordNet. It is freely available for non-profit research purpose at http://sentiwordnet.isti.cnr.it/.

SenticNet is a popular knowledge base of concepts (compiled based on commonsense), along with their polarity scores. It provides the semantics and polarity scores associated with about 30,000 multiword expressions (concepts); its current version, SenticNet5, contains 100,000 concepts by adding conceptual primitives. http://github.com/senticnet

MPQA Subjectivity Lexicon The MPQA lexicon [42] includes 8,222 words with their measure of subjectivity (strong or weak), part-of-speech tags, and their polarities. http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/.

Harvard General Inquirer Harvard General Inquirer [43] contains 182 categories including positive and negative marking for 1,915 positive words and 2,291 negative words. http://www.wjh.harvard.edu/~inquirer/.

Hu and Liu Opinion Lexicon This lexicon [44] includes 4783 negative words and 2006 positive words. https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html.

4 Aspect-Level Sentiment Analysis

Aspect-level sentiment analysis captures opinions per opinion targets (which is considered as "aspect"). This granularity allows, for example, comparing products according to their specific aspects (e.g., a laptop's screen or price), instead of extracting opinions about a whole product (e.g., laptop). There are two major challenges in aspect-level SA. The first relates to the recognition that several opinions can co-occur in the same text fragment, for multiple aspects. Consider the following text: "the recovery was fast but the stuff was unfriendly." A positive viewpoint is conveyed for *recovery*, while a negative viewpoint for *staff*. The second challenge stems from the ambiguity of some sentiment words, which acquire context-sensitive polarity. Consider the word "cold" that conveys a positive sentiment in *cold beer* and negative in *cold pizza*.

Because natural languages do not always comply with simple rules, addressing the first challenge that involves connecting aspects with sentiment words is not a simple task. In the excerpt "the staff is caring and also very gentle but the pain I feel is intensive," applying a rule that connects adjectives with their nearest noun will identify *pain* as being *gentle*. Thus, connecting sentiment words with their corresponding aspects becomes crucial for aspect-level sentiment analysis. This can be addressed by extraction methods using patterns, dependency parsers, or by designated classification tasks [8]. When a strict approach is taken, i.e., using rigid

pattern structures, accuracy is increased on the expense of coverage, since in last cases a relationship between sentiment word and an aspect is captured.

Another line of approaches do not model the relationship between aspect and sentiment words, but instead fuse context information into the representation of the sentence; attention-based long short-term memory (LSTM) network can account the aspect during attention in two main ways: one way involves concatenating the aspect vector into the sentence hidden representation for computing attention weights, and the other way is appending the aspect vector into the input of every word vector. When the relationship between aspect and sentiment words is modeled, as shown by the next algorithms Ex1 and Ex2, it is more likely to obtain more accuracy on the expense of coverage and to yield explainable results. On the other hand, the method given by Section 4.2 does not model the relationship, but likely to attain relatively high coverage. Pragmatically, several methods could be combined in a cascade approach, while the most accurate method is employed first, and if there is no solution (e.g., the relationship between the aspect and the sentiment words could not be recognized), another method is adopted.

Aspect Detection The aspect detection phase is a preliminary phase that aims to identify aspects that are in focus for sentiment computation. This includes identifying implicit and explicit aspects and normalizing them to canonical form; however, this phase is beyond the scope of this work, since there are many other methods that perform the task in a satisfactory manner [45, 46]. Note that this step becomes redundant if the desired aspects are known. For more reading, please refer to [47].

4.1 Unsupervised

Supervised methods rely on the availability of labeled data; however, because it is impractical to obtain labeled data for every aspect (e.g., screen, price, weight, battery life, etc.), the unsupervised paradigm is more suitable for aspect-level SA. In this line of work, it is considered a logical choice to use adjectives as sentiment words [48, 49, 50], since their function is to characterize nouns. Without being restricted to adjectives, Blair-Goldensohn et al. [51] report that adjectives comprise 90% of their sentiment words, although other semantic groups, as pronouns [52], can convey sentiment.

Below we suggest two corpus-based algorithms that construct a quality sentiment lexicon for each aspect, which comprises adjectives, along with a measure of their polarity to indicate the direction the word deviates from the neutral form. The main advantage of the proposed algorithms is threefold: (1) they utilize only unlabeled data; no labels or metadata are required; (2) they allow easy adaptation to other domains and languages, and (3) they maintain relatively accurate polarity scores for lexical words. These three contributions are ascribed primarily to the following properties of the algorithms: a. An exhaustive corpus-based expansion

process that opens up the ability to use only two seed sentiment words ("good" and "bad") and two conjunction patterns ("and" and "but") that are trivial to obtain in many languages. b. Using adjectives as sentiment words, a choice that stems from their functionality to directly convey information on nouns. c. Minor reliance on dependency parsing. Recall that parsers are shown to be less effective in processing informal or transcribed text [14]. The following methods use a parser mainly to identify relations between nouns and adjectives. Being a sub-task of a dependency parser, identifying such relations can be solved by a classification process, as suggested in [8].

Ex1: Algorithm for Lexicon Construction The purpose of the Ex1 algorithm [40] is to construct a lexicon of sentiment words, in this case adjectives, given a target aspect a, and to assign to each adjective a polarity measure in the context of a.

Connecting Aspects with Adjectives For each aspect a, given a sentence s, we seek to identify the set of adjectives in s that are semantically related to a and to associate them with aspect a. For this association, the method to connect adjectives with nouns in classification, presented in [8], can be employed. Alternatively, a dependency parser—capable of producing dependency representation in order to provide a simple description of the grammatical relationships in a sentence—can be employed. It outputs a set of triplets, each comprised a pair of words, and a label to represent their dependency (or relation) type. For example, the noun *service* is recognized by the Stanford Parser[1] to be modified by the adjective *excellent* in the following review taken from TripAdvisor: "Breakfast at the hotel is very good and the service of the staff here is excellent." This is given by the triplet <amod: service, excellent>. The question whether to use a dependency parser or a classifier depends on the nature of the data (e.g., level of noise) and the availability of a high-quality parser in the target language; for further discussion and more examples, refer to [8].

We consider aspect a and adjective jj connected if:

Definition: connected: Aspect a and adjective jj that co-occur in the same sentence are connected if they comply with three constraints: (1) they are interdependent, and the dependency type is one of the following $\{amod, nsubj, dep\}$, where $amod$ captures adjectival modifier, $nsubj$ captures a noun phrase that is the syntactic subject of a clause, and dep is an unlabeled dependency. (2) The dependency governor is the aspect a, and its POS tag is either NN or NNS (singular or plural noun). (3) The dependency dependent is the adjective jj, and its POS tag is JJ, JJR (adjectives with the comparative ending) or JJS (adjectives with the superlative ending). When the parser outputs the dep dependency type, it is in cases when it detects a dependency but fails to determine the correct type of the dependency. Therefore, allegedly, it is counterproductive to use dep. However, due to the informal language in UGC, there are many cases in which the dependency type is not recognized (e.g., in "staff: very helpful"), and therefore, considering

[1] http://nlp.stanford.edu:8080/parser/index.jsp.

this dependency type is important. When there are two adjectives jj_1, jj_2 in the sentence, while the first adjective (jj_1) is connected with aspect a, and there is a conjunct relation between the two adjectives, we also consider *a connected* with jj_2. The conjunct relation indicates a coordinating conjunction dependency type and implies that the second adjective (jj_2) is also modifying aspect a. Consider the following excerpt: "the room is wide and clean." The adjective "wide" is connected with the aspect "room" according to the above-mentioned constraints. Since there is a conjunct relation in between the adjectives "wide" and "clean," the adjective "clean" also becomes *connected* with the aspect room.

Based on the *homogeneous condition* that constrains two arguments that share something in common [53], two types of conjunction patterns are defined.

Definition: Homogeneous pattern: The "and" conjunction is defined as a *homogeneous pattern* since it connects two adjectives that share semantic meaning in common; we consider it semantic orientation feature.

Definition: Diversified pattern: The "but" conjunction is defined as a *diversified pattern* since it connects two adjectives that have contradictory features; we consider this contradictory feature to be the semantic orientation.

These patterns can be easily obtained in most languages. Next, we define two types of interactions that constrain the semantic orientation of their arguments.

Definition: Interaction: Two adjectives jj_1, jj_2 that co-occur *interact* with each other if they are both *connected* with the same occurrence of aspect a.

Definition: Homogeneous interaction: Two adjectives are in *homogeneous interaction* if they adhere to the following constraints: (1) there is an *interaction* involving the two adjectives and (2) there is a *homogeneous pattern* between the two adjectives, and either none of them is negated, or both of them are negated; or there is a *diversified pattern* between the two adjectives, while only one of them is negated. For example, there is a *homogeneous interaction* in the sentence "the pool is cold and small," between the adjectives *cold* and *small*, because both adjectives are *connected* with the aspect "pool," and there is a *homogeneous pattern* between the adjectives, while none of them is negated.

Definition: Diversified interaction: Two adjectives are in *diversified interaction* if they adhere to the following constraints: (1) there is an *interaction* involving the two adjectives and (2) there is a *diversified pattern* between the adjectives, and either none of them is negated, or both of them are negated; or there is a *homogeneous pattern* between the two adjectives, while only one of them is negated. Consider the sentence "the pool is not crowded and large," where both adjectives, "crowded" and "large," are "connected" with the aspect "pool." In addition, there is a *homogeneous pattern* between the adjectives, and one of them is negated, and therefore, it is a *diversified interaction*.

For negation detection, a set of prefixes can be used and the indication of the dependency parser. This also facilitates the process of pattern matching; when looking for *homogeneous* or *diversified patterns*, we generalize negation indicators in the following way: a negated adjective jj is transformed to the form of $\neg jj$, and the negation indicator is omitted from the text. For example, after detecting negation in the sentence "the pool is crowded and not large," it is transformed to the

Algorithm 1 Ex1: Constructing Lexicon for aspect a

Input: SL – seed lexicon $\left\{jj\text{: polarity}|\forall jj,jj.\text{polarity} = \left\{1,\text{if positive} \quad 0,\text{if negative}\right\}\right\}$
Input: D – corpus of reviews, segmented to sentences $\{s_1, \ldots, s_n\}$
Input: P – set of patterns
Output: *lexicon(a)* – the extended lexicon for aspect a
1: $M \leftarrow \emptyset$; $itr \leftarrow 0$; Initialize *DAG(a)* with SL
2: Fill M with all interactions of adjective pairs in D which are connected with aspect a;
3: $itr \leftarrow itr + 1$;
4: Let jj_x^i be adjective x, where i represents the iteration (itr) in which it was added to $DAG(a)$ adjective $jj_x^i \in \mathcal{D}AG(a)$ **where** $i == itr - 1$ interaction in M comprised of jj_x^i and jj_y^j **where** ($jj_y^j \notin \mathcal{D}AG(a)$ **or** $j == itr$)
5: $neg(jj_x^i) \leftarrow$ detect negation of jj_x^i; $neg(jj_y^j) \leftarrow$ detect negation of jj_y^j
6: $p_k \leftarrow$ extract pattern between jj_x^i and jj_y^j detected homogeneous or diversified interaction using $neg(jj_x^i)$,$neg(jj_y^j)$,p_k
7: Add jj_y^j to $DAG(a)$ **unless** $jj_y^j \in \mathcal{D}AG(a)$
8: Update edge from jj_x^i to jj_y^j with counter corresponds to the interaction type $jj_y^j \in \mathcal{D}AG(a)$, **where** ($j == itr$)
9: jj_x^i.polarity $\leftarrow$ compute polarity using incoming edges information no adjective was added to DAG(a) in current iteration (*itr*)
10: **Return** a lexicon of all adjectives in $DAG(a)$

form "the pool is crowded and $\neg large$" and the inverse sentiment score of "large" is considered; thereby, the "and" pattern can be matched in between the adjectives, and subsequently, *diversified interaction* is detected.

The exhaustive algorithm generates a graph of adjectives for every target aspect a, *DAG(a)*, and computes the polarity for each adjective. Algorithm 1 provides a detailed description for constructing the lexicon. The algorithm's main steps are described below:

Step 1 *Initializing*. At starting point, add to *DAG(a)*: *good*, which is assigned with polarity score "1" (represents positive polarity), and *bad*, which is assigned with polarity score "0" (represents negative polarity). For each iteration i, using corpus D:

Step 2 *Graph expansion*. Iterate through all occurrences of each $jj_1 \in \mathcal{D}AG(a)$ in D, where jj_1 is *connected* with a, and count the number of *homogeneous* and *diversified* interactions with each adjective $jj_2 \notin \mathcal{D}AG(a)$. Add adjective jj_2 to *DAG(a)* at the end of this step with the incoming edges from *interacting* adjectives, and record interaction type counters. For example, consider the sentence "the pool is crowded but wide." Assume: a='pool,' i=2, 'crowded' $\notin \mathcal{D}AG(a)$, 'wide' $\in \mathcal{D}AG(a)$. Since there is a *diversified interaction* of the adjectives "crowded" and "wide," "crowded" will be added to the graph with an incoming edge from the node "wide." Figure 1 illustrates the graph at the end of iteration $i = 2$. It can be seen that the adjective "crowded" has sixteen interactions in the corpus with the

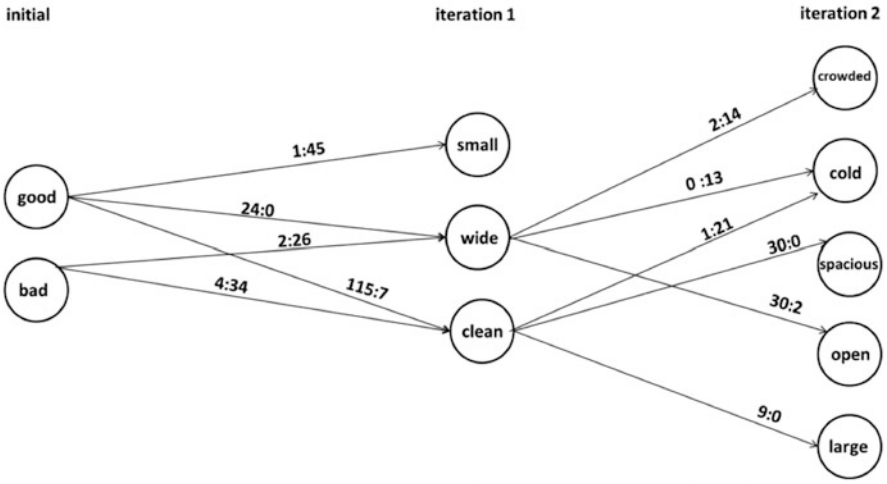


Fig. 1 Example of adjective graph for the aspect pool. Labels above each edge represent homogeneous (left) and diversified (right) counters

adjective "wide," and in fourteen out of sixteen times, the interaction type is *diversified*.

Step 3 *Polarity assignment*. At the end of the iteration, the polarity of new adjectives that were added to the graph in the current iteration is computed. This is performed by averaging the polarities of adjectives from which edges are incoming; the average is weighted according to the number of interactions of each adjective. In the case of diversified interaction, consider the inverse polarity of the source adjective (i.e., 1-polarity). The final polarity of each adjective is in the range of [0:1], since the initial polarity assignment of seed adjectives is either "0" or "1." Once an adjective is added to the graph, it is used in extracting new adjectives and its polarity score is clamped.

Steps 2 and 3 repeat until no new adjective is added to $DAG(a)$.

Ex2: Lexicon Construction Through Co-Expansion. The co-expansion algorithm utilizes a mutual expansion process—patterns are used to expand the sentiment word lexicon and vice versa, and sentiment words are used to expand the extraction patterns set—thus its name.

Definition: Interaction: An occurrence of two adjectives jj_1, jj_2 that *interact* with each other if they are both *connected* with same occurrence of aspect a.

After identifying *interacting* adjectives, patterns in between them can be extracted. Based on the *homogeneous condition* that constrains two arguments that share something in common, two classes of patterns are defined by utilizing the extracting adjectives' polarity. If a negated adjective is detected, the inverse polarity of the adjective is counted.

Each extracted pattern pertains to one of the following classes:

Definition: Homogeneous pattern: A set of consequent words in between two *interacting* adjectives is a *homogeneous pattern* when the adjectives share something in common; we consider this homogeneous feature to be semantically oriented, i.e., both adjectives are either positive or negative. Consider the excerpt "the room is wide and very clean." Since the adjectives "wide" and "clean" convey positive-semantic orientation in this context, the pattern "and very" becomes *homogeneous pattern*.

Definition: Diversified pattern: A set of consequent words in between two interacting adjectives is a *diversified pattern* when the adjectives have contradictory features; we consider this contradictory feature to opposing orientation class, i.e., one is positive and the other is negative. Consider the following sentence involving two interacting adjectives: "the room is outdated but quite spacious." In this context, the polarity of the adjective "outdated" is negative, and the polarity of the adjective "spacious" is positive. Due to the polarity of their semantic orientations, the pattern "but quite" becomes *diversified pattern*.

Once patterns are extracted and their function is determined, they are used to extract new adjectives and compute their polarity, because they constrain the semantic orientation of their arguments.

The process is presented in Algorithm 2, and its main steps are as follows:

Step 1: *Initializing*. At the starting point, the two adjectives: "good," which is assigned positive polarity, and "bad," which is assigned negative polarity, are added to *DAG(a)*; the two patterns: "and," which is considered a *homogeneous pattern*, and "but," which is considered a *diversified pattern*, are added to the pattern set P. Seed adjectives' polarities and seed patterns' classes are considered statistically significant. For each iteration i, using corpus D:

Step 2: *Adjective extraction*. Use statistically significant patterns (step 5) from P and statistically significant adjectives (step 3) from *DAG(a)* to extract new adjectives. This begins by iterating through all *interactions* in D. Extract adjective $jj_2 \notin \mathcal{D}AG(a)$ if it interacts with statistically significant $jj_1 \in \mathcal{D}AG(a)$, and the pattern in between them p is known ($p \in \mathcal{P}$) and statistically significant. Add jj_2 into *DAG(a)* at the end of this step with the incoming edge from jj_1, and retain counters for encountered patterns classes. Negation of adjectives is considered, that is, if there is an odd number of negated adjectives, i.e., one adjective is negated, the other pattern class counter is incremented.

Consider the sentence "the pool is crowded but pretty wide." Given: a='pool', i=2, crowded $\notin \mathcal{D}AG(a)$, wide $\in \mathcal{D}AG(pool)$ (and is significantly positive), "but pretty" $\in \mathcal{P}$ (and is significantly diversified). Since the adjectives crowded and wide interact, and the pattern in between p is "but pretty," the adjective "crowded" is added to *DAG(pool)* with the incoming edge from the node "wide." The *diversified* counter for this edge is incremented. If one of the adjectives was negated, the *homogeneous* counter would be incremented. Figure 1 illustrates the graph at the end

of iteration $i = 2$. It can be seen that the adjective *crowded* had been extracted by using the adjective *wide*; the *diversified* counter has been incremented fourteen out of sixteen possible times.

Step 3: *Polarity assignment by statistical test*. After processing the entire dataset, a two-tailed test is conducted for each adjective in *DAG(a)* that is not marked statistically significant. The null hypothesis is tested to determine whether the adjective's polarity is drawn from a random distribution. The adjective population is comprised of polarity values—either positive or negative—of the incoming edges' nodes, according to their counters. The random population is drawn from all of the extraction events of all of the adjectives in *DAG(a)*, i.e., polarity values of all nodes that have outgoing edges, according to their counters. In cases of *diversified* patterns, the inverse polarity of the adjective is considered. If the difference is found to be statistically significant, we clamp the adjective class, positive (equivalent to "1") or negative (equivalent to "0"), which is determined according to the *p-value*. Otherwise, the adjective is not used in the co-expansion process due to the uncertainty with regard to its polarity. Nevertheless, they are included in the lexicon and can be utilized in sentiment detection tasks, while their polarity score is set to the *p-value*. For example, when creating the population of the node "wide," we consider all of its incoming edges (see Fig. 1). The population is 26 times the value positive, since it has 24 *homogeneous interactions* with "good" (positive polarity) and two *diversified interactions* with "bad" (negative polarity, which is the inverse polarity of positive), and two times the value of negative due to the *homogeneous interactions* with "bad" (negative polarity). It can be seen that the polarity of "wide" is marked statistically significant in the first iteration, since it is used in expanding the graph (i.e., it has outgoing edges). The motivation behind the statistical tests is to ensure the quality of the extraction process. That is, only adjectives whose polarity values are not drawn from a random distribution are used in co-expansion. This increases the precision at the expense of recall, since not all adjectives are used in extracting new adjectives. However, by mining the entire dataset in each iteration, we wish to increase recall by constantly adding new adjectives to the graph (until exhaustion).

Step 4: *Pattern extraction*. Find all interactions $< jj_1, jj_2 >$ in D where $jj_1, jj_2 \in \mathcal{DAG}(a)$ and both adjectives are statistically significant. Extract the pattern p in between each such adjective pair and add to P. If jj_1 and jj_2 have the same polarity (after considering negation), mark p as a *homogeneous pattern*; otherwise, mark it as a *diversified pattern*. At the end of this step, each extracted pattern retains two counters corresponding with each pattern class.

Step 5: *Determine pattern class by statistical test*. The goal of this step is to determine the pattern's class, i.e., *homogeneous* or *diversified*, by conducting a statistical test. Similar to step 3, a two-tailed statistical test is conducted for each statistically non-significant pattern in P. The

null hypothesis is tested to determine whether the pattern's class is drawn from a random distribution. A pattern's population is comprised of *homogeneous* and *diversified* values multiplied by their counters, according to step 4. The random population is drawn from all of the counters of patterns in P, i.e., classes of all patterns multiplied by their counters. If the difference is found to be statistically significant, the pattern is marked as either *homogeneous* or *diversified* according to the *p-value*; otherwise, the pattern is not used in the co-expansion process due to the uncertainty with regard to its class. The test is performed for all statistically non-significant patterns, whether extracted in the current iteration or in previous iterations; recall that the pattern set is constantly growing due to step 4, and therefore, the random distribution is also constantly changing. We keep extracting patterns (step 4) and updating their counters until they are found to be significantly *homogeneous* or *diversified*.

Steps 2–5 repeat until no adjective polarity or pattern class is found to be statistically significant.

Once aspect lexicons are constructed, they are used in computing sentiment by the *connected* principle, and by presenting the supporting adjectives, the computation is explainable. Because those methods are designed to achieve high accuracy, in the expense of coverage, other methods can be used in a fallback approach; if, for example, not a single known adjective is *connected* with the target aspect, another method can be employed.

4.2 Weakly Supervised

LARA Latent aspect rating analysis (LARA) [54] is a weakly supervised technique based on rating regression approach. It utilizes the overall rating of each review to predict the latent rating of aspects in the review, and therefore, it is bound to the type of opinioned text that is associated with overall user ratings. The principal of LARA is to train a generative latent rating regression (LRR) model to predict aspect ratings based on the review text and the associated overall rating; therefore, it is considered to be a weakly supervised technique. LRR assumes that the overall rating is generated based on a weighted combination of the latent ratings over all the aspects, where the weights constitute the relative emphasis that the reviewer has placed on each aspect when giving the overall rating. They propose aspect segmentation algorithm that ascribes each sentence to the aspect that shares the maximum term overlapping with it and use this in training their model. More works rely on minimal annotations, such as in [55] and [56].

Algorithm 2 Ex2: Constructing lexicon for aspect a

Input: SL – seed lexicon $\left\{ jj_k\text{:polarity} \mid \forall jj_k, jj_k\text{.polarity} = \left\{\text{positive} \quad \text{negative}\right\}\right\}$
Input: D – corpus of reviews, segmented to sentences $\{s_1, \ldots, s_n\}$
Input: SP – seed set of extraction patterns $\left\{\text{p:class} \mid \forall p,\ p.class = \left\{\text{homogeneous} \quad \text{diversified}\right\}\right\}$
Output: *lexicon(a)* – the extended lexicon for aspect a

1: $M \leftarrow \emptyset$; $itr \leftarrow 0$; Initialize $DAG(a)$ with SL; Initialize P with SP
2: Fill M with all interactions of adjective pairs in D which are connected with aspect a;
3: $itr \leftarrow itr + 1$;
4: Let jj_x^i be adjective x, where i represents the iteration (itr) in which it was added to $DAG(a)$, and its property sig indicates whether it is statistically significant or not; adjective $jj_x^i \in \mathcal{D}AG(a)$ **where** $jj_x^i.sig$==true interaction in M comprised of jj_x^i and jj_y^j **where** ($jj_y^j \notin \mathcal{D}AG(a)$ **or** $j == itr$)
5: $neg(jj_x^i) \leftarrow$ detect negation of jj_x^i; $neg(jj_y^j) \leftarrow$ detect negation of jj_y^j
6: $p_k \leftarrow$ extract pattern between jj_x^iandjj_y^j $p_k \in \mathcal{P}$ and p_k.sig==true
7: Add jj_y^j to $DAG(a)$ **unless** $jj_y^j \in \mathcal{D}AG(a)$
8: Update edge from jj_x^i to jj_y^j and increment pattern class counter using p_k.class, neg(jj_x^i), neg(jj_y^j) $jj_y^j \in \mathcal{D}AG(a)$, where jj_y^j.sig==false
9: jj_y^j.sig $\leftarrow$ perform statistical test using incoming edges information jj_y^j.sig==false
10: jj_y^j.polarity $\leftarrow$ p-value
11: jj_y^j.polarity $\leftarrow$ negative, positive according to p-value interaction in M involving jj_x^i and jj_y^j **where** (jj_x^i, $jj_y^j \in DAG(a)$ and jj_x^i.sig==true and jj_y^j.sig==true)
12: $p_k \leftarrow$ extract pattern between jj_x^iandjj_y^j **not** ($p_k \in \mathcal{P}$ and p_k.sig==true)
13: Add $p_k \notin \mathcal{P}$ to P
14: Update pattern class counters using jj_x^i.polarity, jj_y^j.polarity, neg(jj_x^i), neg(jj_y^j) $p_k \in \mathcal{P}$ **where** p_k.sig==false
15: $p_k.sig \leftarrow$ perform statistical test using counters p_k.sig==true
16: $p_k.class \leftarrow$ {homogeneous, diversified} according to p-value no adjective of pattern were found statistically significant in current iteration (itr)
17: **Return** a lexicon of all adjectives in $DAG(a)$

4.3 Supervised

In this context, supervised methods rely on the availability of a measure of polarity of each target aspect in each sentence. The availability of such data attracted attempts to incorporate aspect information in various of deep neural network architectures.

AdaRNN AdaRNN [57] adopts recurrent neural network (RNN) to utilize the dependency tree structure. It manipulates the dependency tree based on the aspect target and then performs semantic compositions of words in bottom-up manner. The target-oriented tree helps in propagating sentiment information forward, while the information is dependent on the target. As we travel up, the vectors of two child nodes along with their dependency types are used to select a composition function. The main steps are as follows:

Step 1: Generate a dependency tree of the sentence containing the target, by using dependency parser.
Step 2: Convert the dependency tree recursively, starting with the target node, and finding all words connected to the target. These words will be combined with the target node by a certain order, and the process is recursively employed.
Step 3: The dimensions of a parent node are calculated by a linear combination of the child vectors' dimensions using composition functions. They learn to select a composition function according to the input parameters: the dependency types as a one-hot binary feature vector, along with the children vectors.
Step 4: Use the root node representation as input feature for the final softmax classifier, to predict the final distribution over sentiment classes.

ATAE-LSTM Attention-based LSTM with aspect embedding [58] appends the given aspect embedding with each word embedding as the input of LSTM and uses attention layer above the LSTM layer; the main steps are as follows:

Step 1: Employ LSTM on the input sentence.
Step 2: Regard the last hidden vector N as the representation of the sentence and put N into a softmax layer after linearizing it into a vector whose length equals to the number of class labels.
Standard LSTM cannot effectively detect which is the important part for aspect-level sentiment classification. In order to address this issue, it is proposed to design an attention mechanism that can capture the key part of the sentence in response to a given aspect.
Step 3: To make the best use of aspect information, learn an embedding vector for each aspect.
Step 4: The specific aspect target embedding is incorporated to each hidden state to be considered in computing the attention weight of the word.

Notice that LSTM is very time-consuming during training since it processes one token in a step, and demands a relatively large amount of training set. [59] suggest a traditional machine learning method that uses SVM classifier. It is based on a large amount of features that have been derived mainly from the aspect term itself, lexical features on its surrounding and on its relatives in a parsed tree structure.

4.4 Resources

TripAdvisor Travel Reviews TripAdvisor provides, in addition to textual reviews, multiple user ratings for each one of them. Each review contains ratings ranging from "1" to "5" stars for seven aspects: value, room, location, cleanliness, check-in/front desk, service, and business services, in addition to an overall rating. Wang et al. [54] prepared this dataset that was collected in a one-month period (from

February 14, 2009 to March 15, 2009) on 1,850 hotels and 239,963 hotel reviews. If this dataset becomes unavailable, the author of this chapter will be happy to share it.

Twitter This dataset was collected by querying the Twitter API with target names, services, and products, such as "bill gates," "taylor swift," "xbox," "windows 7," "google." After obtaining them, nearly 7,000 tweets were manually labeled with sentiment labels (negative, neutral, positive) for these targets. In order to eliminate the effects of data imbalance problem, they randomly sample the tweets and make the data balanced. The negative, neutral, and positive classes account for 25%, 50%, and 25%, respectively. http://goo.gl/5Enpu7.

Amazon Laptop Reviews A collection of reviews posted on Amazon.com from June 1995 to March 2013.

Yelp Restaurant Reviews This dataset contains customer reviews posted on the Yelp website. The businesses for which the reviews are posted are classified into about 500 categories, and many of the businesses are assigned multiple business categories.

5 Sentence-Level Sentiment Analysis

In this chapter, we regard a sentence as any fragment of text—a short set of consecutive written words. This can be, for example, a sentence in a review or a tweet. A straightforward approach for sentence-level sentiment analysis involves looking on sentiment words themselves. However, from a linguistic viewpoint, ignoring words order in semantic task could overlook important contextual information. For example, the word "blue" has different meanings in the context of *lips* and in the context of *sky*.

Typically, the supervised machine learning approach is adopted for sentence-level SA that relies on the availability of labeled data. Apart from obtaining a training set by manual annotation, or exploiting labeled sources, distant supervision labels the data automatically in cases when it is straightforward to do so; in Twitter, distant supervision is popular by using emoticons to determine the sentiment class of tweets [60]. After obtaining a training set, typical machine learning approaches require text representation, while it is challenging to go beyond lexical information.

Deep learning approaches alleviate the representation challenge by learning it according to the task. Despite that these approaches typically require a large number of labeled samples and their results can be challenging to explain, they attract many researchers to design mainly variations of LSTM and CNN architectures. Wang et al. [61] design a joint CNN and RNN architecture for sentiment classification of short texts, which takes advantage of the coarse-grained local features generated by CNN and long-distance dependencies learned via RNN. The work of [62] developed the Stanford Sentiment Treebank dataset. They proposed recursive neural tensor

network for capturing the interaction among the words in a sentence. A phrase is represented by word vectors and a parsing tree; vectors for higher nodes in the tree are computed by the same tensor-based composition function.

conSent conSent [63] is a supervised classification method that requires positive and negative texts. It does not rely on sequential information and is tolerant to texts that do not adhere to the rules of grammar and to texts that involve missing words. Instead, it builds on techniques from the field of information retrieval to identify key terms indicative of the existence of sentiment. Additionally, the contexts in which they appear are analyzed and used to generate features for supervised learning. This allows to fuse features from multiple sources to the feature set. In the paper, the robustness to noisy data is demonstrated in Twitter, transcribed text, and reviews.

Key Terms Identification

Step 1: Generate a language model for the set of positive-sentiment documents (T_{pos}).
Step 2: Generate a language model for the set of negative-sentiment documents (T_{neg}).
Step 3: For each term t result in step 1, compute *score(t)* that is the ratio of its frequency in positive-sentiment and negative-sentiment documents.
Step 4: If *score(t)* exceeds a predefined threshold, t is identified as a key term.

Context Terms Identification. For each key term t:

Step 1: Locate all instances of t both in T_{pos} and in T_{neg}. If a document contains several instances of the same key term, the following steps are applied for each instance separately.
Step 2: For each instance found in T_{pos}, extract the terms located around t by using a sliding window of size X denoted as the context span. Every term $t_{context}$ in this text excerpt, aside from the key term itself, is considered a possible context term.
Step 3: Group all context spans of T_{pos}, and generate a language model. Repeat the process for T_{neg}.
Step 4: Subtract for each $t_{context}$ its score in T_{pos} language model from that of T_{neg}.
Step 5: If score($t_{context}$) exceeds a predefined threshold, it will be defined as a context term for the key term t.

Based on key terms that were identified, and on their context terms, various of features are computed. For example, they compute the number of key terms that share at least one context term in the analyzed text. Finally, a classifier is trained to discriminate between positive and negative examples, in this case the rotation forest [64].

CharSCNN A Character to Sentence Convolutional Neural Networks (CharSCNNs) [65] model is proposed. CharSCNN uses two convolutional layers to extract relevant features from words and sentences of any size to capture sentiment information. Following are the main steps:

Step 1: Word-Level Embeddings. For capturing syntactic and semantic information, word-level embeddings are utilized to learn embedding matrix of all words in the vocabulary; in the proposed study, the English Wikipedia is taken as a source of unlabeled data for learning the embeddings.

Step 2: Character-level embeddings. The motivation here is to extract features around each character of the word and, specifically, to capture morphological and shape information, so as Twitter tokens such as #SoSad could be utilized. The convolutional layer applies a matrix–vector operation to each window of characters; then, they are combined using a max operation to create a fixed-sized character-level embedding of the word. The convolutional layer allows the extraction of relevant features from any part of the word and does not need handcrafted inputs such as stems and morpheme lists. As opposed to word embeddings, character-level embeddings are not pre-trained, but instead initialized randomly by sampling from a uniform distribution.

Step 3: Sentence-Level Representation. In this step, a second convolutional layer is used to compute the sentence-wide feature vector in a similar way it was used for character level. The input is a joint word-level and character-level embedding, and the outputs are local features around each word in the sentence; then they are combined by a max operation to form a fixed-sized feature vector representation.

Step 4: Classification. The network is trained to perform sentiment classification task by minimizing a negative likelihood over the training set.

SA of Twitter Data In their study, Agarwal et al. [66] explore several feature groups as well as utilize tree representation of a tweet. The first feature model uses several feature types, including lexical and POS features. They found that the most important features are those that combine prior polarity of words (using dictionaries) with their POS tags, while other features only play a marginal role. A tree model is designed to represent tweets, combining many categories of features in one succinct convenient representation. This is done by adapting partial tree kernel algorithm [67] to calculate the similarity between sub-trees of tweet. Finally, it is shown that combining unigrams with the best set of features outperforms the tree kernel-based model and gives about 4% absolute gain over a unigram baseline.

5.1 Resources

Stanford Sentiment Treebank This dataset is suggested as a benchmark for sentiment analysis. It comprises 11,855 sentences taken from the movie review site Rotten Tomatoes. Every sentence in the dataset has a label that goes from very negative to very positive in the scale from 0 to 1. The labels are generated by human annotators using Amazon Mechanical Turk. Every sentence is parsed to phrases that

were also manually annotated in this scale; in total, fine-grained sentiment labels are given for 215,154 phrases. http://nlp.Stanford.edu/sentiment/.

CSN People diagnosed with cancer, as well as caregivers for individuals with cancer, join the Cancer Survivors Network (CSN) of the American Cancer Society to seek social support and information from members who have experienced a particular situation first hand, as well as to seek emotional support. More than 468,000 forum posts from 48,779 threads were downloaded from the Cancer Survivor Network, posted from July 2000 to October 2010 by 27,173 participants. In total, 293 posts were selected randomly from the breast cancer forum of the Cancer Survivor Network, and each post was manually classified as being of positive or negative sentiment, with the result that 201 of them were labeled as positive and 92 were labeled as negative. Further details are given in [68].

IMDB The IMDB dataset consists of 100,000 movie reviews taken from IMDB given two types of labels: positive and negative. These labels are balanced in both the training and test sets. http://ai.Stanford.edu/amaas/data/sentiment/index.html.

Stanford Twitter Sentiment This dataset was collected by [60], which contains 1.6 million automatically tagged tweets (half-positive and half-negative); they serve as the training set and 359 manually tagged tweets that serve as the test set. http://help.sentiment140.com/.

6 Applications

Online Health Communities Due to the impact that social networks have on individuals, it is encouraged to utilize sentiment analysis knowledge in supporting the lives of millions of online users. Specifically, there is a potential to help people in Online Health Communities—a resource that provides an outlet for patients and caregivers to discuss their related issues. Studies of online cancer support groups and communities have shown that members benefit from online interactions in multiple ways: increased optimism [69], reduced stress, depression, psychological trauma [70], and reduced cancer concerns [71]. People typically join online communities to get information and support but quickly discover that giving support to others is equally important to their survivorship [72].

Portier et al. [1] address community support within the Cancer Survivors Network (CSN) community by assessing the change in sentiment between an initiating post and the first follow-up post of the initiator. Their goal is to examine whether sentiment change, a measure of social support, is influenced by the main topic of the initiating post. Another application is enabling investigation of factors that affect the sentiment change and discovery of sentiment change patterns, by analyzing Online Health Communities [68].

User Reviews Online opinions and recommendations have become a significant decision-making factor in a wide variety of areas. Many people rely on them when planning their next vacation, purchasing a new camera, or applying for a new job. However, with the advent of sentiment analysis, an individual is no longer strictly limited to asking friends and family for their opinions; organizations are no longer limited to conducting surveys, opinion polls, and focus groups to sound out public or consumer opinions. Sentiment analysis paves the way to several and interesting applications, in almost every possible domain.

TripAdvisor is a major hotel rating site; due to its popularity, it contains a large number of reviews. Apart from textual content, TripAdvisor.com provides multiple user ratings within each review. Each review contains ratings ranging from "1" to "5" stars for seven aspects: value, room, location, cleanliness, check-in/front desk, service, and business services, in addition to an overall rating. The works of [40], and [54] suggest aspect-level sentiment analysis to enable summarizing reviews and ranking hotels based on their inferred ratings on each different aspect. Profiling users' rating behavior can support a recommendation system in which recommendations are based on reviews of users that share similar rating patterns. In case the user provides outlier rating, the system can ask her if this is a mistake. Automatically surveying hotels by accessing user ratings summarization can help cities in monitoring and improving their hotel services.

More applications belong to politics [73, 74], disaster response [75], financial market prediction [76, 77], and much more.

7 Summary and Discussion

This chapter provides understanding of the sentiment problem in social media platforms. The lexical methods could be the principal components of every sentiment task, while the main challenge is to maintain a quality lexicon, using a large corpus or existing lexical sources. Once constructed, using predefined lexicons in various of sentiment tasks has two main advantages; first, there is no need to annotate further data, and second, since the lexicons are defined independently of the data, it may prevent overfitting. The main strength of the lexicon-based approaches is also their weakness: as the lexicons are predefined, they are unable to adapt to novel or domain-specific forms of expression.

Lexicons are far from being sufficient for sentiment analysis. Specifically, for aspect-level SA, a general sentiment lexicon will fail in adapting to aspects' context. This chapter puts focus on aspect-level SA, which has a great potential to capture multiple opinions, linking them with various of topics, and conduct more in-depth analysis. In this context, the unsupervised methods have coverage problem since they rely on adjectives; other presented methods consider all word classes to convey sentiment; however, this occurs at the expense of accuracy since they do not strictly model the relation between sentiment words and the target aspect.

The end of this chapter explores sentence-level SA, which is similar to document-level SA, since a document can be considered as a collection of (ordinal) sentences. This level of analysis (e.g., a tweet, or a forum post) enables investigating sentiment change in an online community. The conSent method is in focus due to its robustness to noise and its ability to work well in various of social media platforms.

The increasing emergence of deep learning approaches had been proven to function well in various of NLP tasks [9]. The main advantage of employing deep learning approach lies in its independence from expert knowledge, since the text representation is learnt according to the task. However, despite the fact that neural networks have a large capacity to learn complex decision functions, they tend to easily overfit especially on small-sized datasets. Neural models achieve high accuracy in sentiment analysis, specifically in aspect level [78]; despite the attention such methods get, the following must be said. Many of the published methods are not specific for sentiment analysis and work well with many text classification tasks. Therefore, some of the authors do not deeply understand the challenges and complexity of the task, and it feels that their only contribution is in adapting a variation of known deep learning architecture for SA. In this context, state-of-the-art baselines are not considered, error analysis is missing, many works do not discuss their method's strengths and weaknesses, and they lack motivation. This limits the merit of these works and promotes poor understanding of the problem.

Notice that the ATAE-LSTM method in Sect. 4 outperforms standard LSTM, which cannot capture any aspect information, only by nearly 2% in all evaluated datasets. Similar observation is given for the AdaRNN method on their Twitter dataset; using a simple RNN on the converted dependency tree yielded 63% accuracy, comparing with 66.3% for AdaRNN. This may tell us that most sentiment information can be captured by methods that ignore the target aspect (e.g., standard LSTM); could it be that the majority of the sentences do not contain shift in sentiment but rather have homogeneous polarity?

In the study of [79], it is reported an average drop in performance of nearly 20% for sentences that have multiple sentiments and multiple aspect terms (hard cases). This invites further discussion on the merit of complex methods, and performing evaluation on hard cases. Apart, we encourage researchers to provide examples and error analysis; this is essential to understand in which cases the method was successful, and why.

7.1 A Call for Responsible SA

Artificial intelligence (AI) is bringing about profound changes in the lives of human beings, and it will continue to do so. The coronavirus era demonstrated this by showing us how online communications, social support, and e-learning can be done by using AI-based technologies. We are witnessing a world in which many decisions are not made by humans anymore; self-driving cars, algo-trading, and fraud detection well demonstrate this.

Sentiment analysis has the potential to open the doors to investigate how people influence each other, what are their opinions, and why. This invites us to discuss in every research project privacy concerns. The kind of technology we create provides organizations and authorities with valuable knowledge, while the benefit of the users is not guaranteed.

The research and development of such praxes invite researchers and practitioners to discuss responsible usage, from individual and social points of views. When the right questions are asked, e.g., does what we do really contribute to our societies, there could be a synthesis between research, praxis, and responsibility. When we are able to see the data we collect and mine as something that was written by people care about, maybe we could be more sensitive to the consequences of our research and applications.

References

1. Kenneth Portier, Greta E Greer, Lior Rokach, Nir Ofek, Yafei Wang, Prakhar Biyani, Mo Yu, Siddhartha Banerjee, Kang Zhao, Prasenjit Mitra, et al. Understanding topics and sentiment in an online cancer survivor community. *Journal of the National Cancer Institute Monographs*, 2013(47):195–198, 2013.
2. Cindy K Chung and James W Pennebaker. Revealing dimensions of thinking in open-ended self-descriptions: An automated meaning extraction method for natural language. *Journal of Research in Personality*, 42(1):96–132, 2008.
3. Jessica Elan Chung and Eni Mustafaraj. Can collective sentiment expressed on Twitter predict political elections? In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.
4. Lewis Mitchell, Morgan R Frank, Kameron Decker Harris, Peter Sheridan Dodds, and Christopher M Danforth. The geography of happiness: Connecting Twitter sentiment and expression, demographics, and objective characteristics of place. *PloS One*, 8(5), 2013.
5. Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326, 2010.
6. Nir Ofek and Asaf Shabtai. Dynamic latent expertise mining in social networks. *IEEE Internet Computing*, 18(5):20–27, 2014.
7. Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.
8. Nir Ofek, Lior Rokach, and Prasenjit Mitra. Methodology for connecting nouns to their modifying adjectives. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 271–284. Springer, 2014.
9. Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.
10. Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.
11. Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.
12. Soo-Min Kim and Eduard Hovy. Determining the sentiment of opinions. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 1367. Association for Computational Linguistics, 2004.

13. Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 417–424. Association for Computational Linguistics, 2002.
14. Nir Ofek, Gilad Katz, Bracha Shapira, and Yedidya Bar-Zev. Sentiment analysis in transcribed utterances. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 27–38. Springer, 2015.
15. Nir Ofek, Lior Rokach, Roni Stern, and Asaf Shabtai. Fast-CBUS: A fast clustering-based undersampling method for addressing the class imbalance problem. *Neurocomputing*, 243:88–102, 2017.
16. Jay M Ponte and W Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, 1998.
17. Alexandra Balahur, Ralf Steinberger, Mijail Kabadjov, Vanni Zavarella, Erik Van Der Goot, Matina Halkia, Bruno Pouliquen, and Jenya Belyaeva. Sentiment analysis in the news. *arXiv preprint arXiv:1309.6202*, 2013.
18. Rui Xia and Chengqing Zong. Exploring the use of word relation features for sentiment classification. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1336–1344. Association for Computational Linguistics, 2010.
19. Paul F Evangelista, Mark J Embrechts, and Boleslaw K Szymanski. Taming the curse of dimensionality in kernels and novelty detection. In *Applied soft computing technologies: The challenge of complexity*, pages 425–438. Springer, 2006.
20. Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *International Work-Conference on Artificial Neural Networks*, pages 758–770. Springer, 2005.
21. J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
22. Nir Ofek, Cornelia Caragea, Lior Rokach, Prakhar Biyani, Prasenjit Mitra, John Yen, Kenneth Portier, and Greta Greer. Improving sentiment analysis in an online cancer survivor community using dynamic sentiment lexicon. In *2013 International Conference on Social Intelligence and Technology*, pages 109–113. IEEE, 2013.
23. Cornelia Caragea, Adrian Silvescu, Saurabh Kataria, Doina Caragea, and Prasenjit Mitra. Classifying scientific publications using abstract features. In *Ninth Symposium of Abstraction, Reformulation, and Approximation*, 2011.
24. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
25. Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.
26. Isaac G Councill, Ryan McDonald, and Leonid Velikovich. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 51–59. Association for Computational Linguistics, 2010.
27. Sergey Goryachev, Margarita Sordo, Qing T Zeng, and Long Ngo. Implementation and evaluation of four different methods of negation detection. Technical report, Technical report, DSG, 2006.
28. Noa P Cruz Díaz and Manuel J Maña López. *Negation and speculation detection*, volume 13. John Benjamins Publishing Company, 2019.
29. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford InfoLab, 1999.
30. Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.
31. Delip Rao and Deepak Ravichandran. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682. Association for Computational Linguistics, 2009.

32. Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. SentiWordNet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.
33. Peter D Turney and Michael L Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346, 2003.
34. Jaap Kamps, Maarten Marx, Robert J Mokken, Maarten De Rijke, et al. Using WordNet to measure semantic orientations of adjectives. In *LREC*, volume 4, pages 1115–1118. Citeseer, 2004.
35. Christiane Fellbaum. WordNet. *The Encyclopedia of Applied Linguistics*, 2012.
36. Yang Li, Quan Pan, Tao Yang, Suhang Wang, Jiliang Tang, and Erik Cambria. Learning word representations for sentiment analysis. *Cognitive Computation*, 9(6):843–851, 2017.
37. Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. Building large-scale Twitter-specific sentiment lexicon: A representation learning approach. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical papers*, pages 172–182, 2014.
38. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
39. Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
40. Nir Ofek, Soujanya Poria, Lior Rokach, Erik Cambria, Amir Hussain, and Asaf Shabtai. Unsupervised commonsense knowledge enrichment for domain-specific sentiment analysis. *Cognitive Computation*, 8(3):467–477, 2016.
41. Pei Ke, Haozhe Ji, Siyang Liu, Xiaoyan Zhu, and Minlie Huang. SentiLARE: Linguistic knowledge enhanced language representation for sentiment analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6975–6988, 2020.
42. Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, 2005.
43. Philip J Stone, Dexter C Dunphy, and Marshall S Smith. The general inquirer: A computer approach to content analysis. M.I.T. Press, 1966.
44. Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177, 2004.
45. Soujanya Poria, Erik Cambria, Lun-Wei Ku, Chen Gui, and Alexander Gelbukh. A rule-based approach to aspect extraction from product reviews. In *Proceedings of the Second Workshop on Natural Language Processing for Social Media (SocialNLP)*, pages 28–37, 2014.
46. Toqir A Rana and Yu-N Cheah. A two-fold rule-based model for aspect extraction. *Expert Systems with Applications*, 89:273–285, 2017.
47. Toqir A Rana and Yu-N Cheah. Aspect extraction in sentiment analysis: comparative analysis and survey. *Artificial Intelligence Review*, 46(4):459–483, 2016.
48. Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Expanding domain sentiment lexicon through double propagation. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.
49. Samuel Brody and Noemie Elhadad. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812. Association for Computational Linguistics, 2010.
50. Yunfang Wu and Miaomiao Wen. Disambiguating dynamic sentiment ambiguous adjectives. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1191–1199. Association for Computational Linguistics, 2010.

51. Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George Reis, and Jeff Reynar. Building a sentiment summarizer for local service reviews. *Proceedings of WWW 2008 Workshop: NLP Challenges in the Information Explosion Era* (2008).
52. Nir Ofek, Lior Rokach, Cornelia Caragea, and John Yen. The importance of pronouns to sentiment analysis: online cancer survivor network case study. In *Proceedings of the 24th International Conference on World Wide Web*, pages 83–84, 2015.
53. Michael Elhadad and Kathleen R McKeown. Generating connectives. In *Proceedings of the 13th Conference on Computational Linguistics-Volume 3*, pages 97–101. Association for Computational Linguistics, 1990.
54. Hongning Wang, Yue Lu, and Chengxiang Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 783–792, 2010.
55. Honglei Zhuang, Fang Guo, Chao Zhang, Liyuan Liu, and Jiawei Han. Joint aspect-sentiment analysis with minimal user guidance. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1241–1250, 2020.
56. Zenun Kastrati, Ali Shariq Imran, and Arianit Kurti. Weakly supervised framework for aspect-based sentiment analysis on students' reviews of MOOCs. *IEEE Access*, 8:106799–106810, 2020.
57. Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive recursive neural network for target-dependent Twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (volume 2: Short papers)*, pages 49–54, 2014.
58. Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, 2016.
59. Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. NRC-Canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442, 2014.
60. Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009.
61. Xingyou Wang, Weijie Jiang, and Zhiyong Luo. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical papers*, pages 2428–2437, 2016.
62. Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.
63. Gilad Katz, Nir Ofek, and Bracha Shapira. Consent: Context-based sentiment analysis. *Knowledge-Based Systems*, 84:162–178, 2015.
64. Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.
65. Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.
66. Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca J Passonneau. Sentiment analysis of Twitter data. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 30–38, 2011.
67. Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning*, pages 318–329. Springer, 2006.
68. Baojun Qiu, Kang Zhao, Prasenjit Mitra, Dinghao Wu, Cornelia Caragea, John Yen, Greta E Greer, and Kenneth Portier. Get online support, feel better–sentiment analysis and dynamics in an online cancer survivor community. In *2011 IEEE Third International Conference on Privacy,*

Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, pages 274–281. IEEE, 2011.
69. Shelly Rodgers and Qimei Chen. Internet community group participation: Psychosocial benefits for women with breast cancer. *Journal of Computer-Mediated Communication*, 10(4):JCMC1047, 2005.
70. Christopher E Beaudoin and Chen-Chao Tao. Modeling the impact of online cancer resources on supporters of cancer patients. *New Media & Society*, 10(2):321–344, 2008.
71. Eunkyung Kim, Jeong Yeob Han, Tae Joon Moon, Bret Shaw, Dhavan V Shah, Fiona M McTavish, and David H Gustafson. The process and effect of supportive message expression and reception in online breast cancer support groups. *Psycho-Oncology*, 21(5):531–540, 2012.
72. Bret R Shaw, Fiona McTavish, Robert Hawkins, David H Gustafson, and Suzanne Pingree. Experiences of women with breast cancer: exchanging social support over the chess computer network. *Journal of Health Communication*, 5(2):135–159, 2000.
73. Martin Haselmayer and Marcelo Jenny. Sentiment analysis of political communication: combining a dictionary approach with crowdcoding. *Quality & Quantity*, 51(6):2623–2646, 2017.
74. Ema Kušen and Mark Strembeck. Politics, sentiments, and misinformation: An analysis of the Twitter discussion on the 2016 Austrian Presidential Elections. *Online Social Networks and Media*, 5:37–50, 2018.
75. J Rexiline Ragini, PM Rubesh Anand, and Vidhyacharan Bhaskar. Big data analytics for disaster response and recovery through sentiment analysis. *International Journal of Information Management*, 42:13–24, 2018.
76. Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.
77. Tushar Rao, Saket Srivastava, et al. Analyzing stock market movements using Twitter sentiment analysis. In: ASONAM'12 Proceedings of the 2012 international conference on advances in social networks analysis and mining (ASONAM 2012), pp 119–123.
78. Hai Ha Do, PWC Prasad, Angelika Maag, and Abeer Alsadoon. Deep learning for aspect-based sentiment analysis: a comparative review. *Expert Systems with Applications*, 118:272–299, 2019.
79. Wei Xue and Tao Li. Aspect based sentiment analysis with gated convolutional networks. *arXiv preprint arXiv:1805.07043*, 2018.

Human Resources-Based Organizational Data Mining (HRODM): Themes, Trends, Focus, Future

Hila Chalutz-Ben Gal

1 Introduction

In recent years, there has been a trend in many organizations toward data-driven decision-making in various aspects of business (Holsapple et al. 2014) with the use of big data in daily activities (Chong and Shi 2015). Many organizations are experiencing a period of transformation as modern businesses both exploit opportunities and face numerous and complex challenges. Today's organizational data mining (hereafter ODM) transformation is a direct result of rapid changes within organizations caused by the combined forces of demographics, globalization, and information technology. Some departments (e.g., human resources) rely on data to execute activities that were traditionally performed in a somewhat intuitive manner. This transformation plays a crucial role in firms' ability to achieve a competitive advantage in today's challenging economy (Kapoor and Sherif 2012; Sparrow 2012; Fulmer and Ployhart 2013). In light of the rapid changes in technology and the environment, traditional organizational metrics have become unsuitable for many situations (Fink 2010; Handa and Garima 2014; Sharif 2015).

ODM is defined as leveraging data mining (hereafter DM) tools and technologies to enhance organizational decision-making process by transforming data into valuable and actionable knowledge to gain a strategic competitive advantage (Nemati and Barko 2002, 2003). ODM domains are wide in scope. Some focus on customer relationship management, customer segmentation, retention and attrition management, risk forecasting, and profitability analysis (Kharb 2019; Meghyasi and Rad 2020). Additional ODM domains include organizational processes and Human Resources data (hereafter HRODM) for improved organizational decision-making.

H. Chalutz-Ben Gal (✉)
School of Industrial Engineering and Management, Afeka Tel Aviv Academic College of Engineering, Tel-Aviv, Israel
e-mail: hilab@afeka.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_36

This chapter focuses on HRODM utilizing data to improve people-related organizational decision-making processes. HRODM is sometimes referred to in terms such as "people analytics," "human capital analytics," or "human resources analytics," among others. Within the ODM domain, HRODM is defined as "the application of sophisticated DM and business analytics techniques to the field of human resources" (Vihari and Rao, p. 1). It is also referred to as quantitative and qualitative data and information management that aims to gain insight and support decision-making processes with regard to managing people in organizations (Fitz-Enz 2000; Handa and Garima 2014; Zhao and Carlton 2015). A third definition pertains to "processes to collect, transform and manage key people related data and documents; to analyze the gathered information using DM models; and to disseminate the analysis results to decision makers for making intelligent decisions" (Kapoor and Sherif, p. 1626).

HRODM has several goals. The first is "to gather and maintain data for predicting short and long term trends in the supply and demands of workers in different industries and occupations and to help global organizations make decisions relating to optimal acquisition, development and retention of their human capital" (Kapoor and Sherif, p. 1627). The second is "to provide an organization with insights for effectively managing employees in order to achieve business goals quickly and efficiently" (Davenport et al. 2010; Hota and Gosh, p. 169). Third, some scholars emphasize that the goal of HRODM is to positively influence the successful execution of an organization's strategy (Heuvel and Bondarouk 2016; Huselid 2015; Kapoor and Sherif 2012; Levenson 2005, 2011; Zang and Ye 2015).

In this chapter, we propose a new definition for the adoption of HRODM by focusing on the return on investment (hereafter ROI) gained by an organization when utilizing HRODM tools. ROI is a performance measure used to evaluate the efficiency of an investment or compare the efficiency of a number of different investments. ROI tries to directly measure the amount of return on a particular investment, relative to the investment's cost. To calculate ROI, the benefit (or return) of an investment is divided by the cost of the investment. The result is expressed as a percentage or a ratio. We propose an ROI-based focus of HRODM, because it enables organizational insights and supports decision-makers with respect to the human capital dilemma by providing business insight and consequently helping them make better business decisions. Our proposed ROI-based approach is grounded upon a systematic review and analysis of the literature in the field. In recent years, the connection between ODM and HR has resulted in a growing body of literature that proposes various approaches to combining the two disciplines, sometimes in an unstructured, blunt manner. Moreover, despite notable evidence of a growing interest in HRODM, researchers have found very limited scientific evidence to help decision-makers determine whether and how to adopt and implement HRODM (Rasmussen and Ulrich 2015).

This chapter aims to bridge this gap by proposing an ROI-based review of HRODM in the sense that the efforts required to adopt analytic data mining methods and apply them to HR tasks must be justified. This chapter has two objectives. The first objective is to provide an integrative analysis of the literature on the

topic of HRODM through the lens of ROI to provide scholars, executives, and practitioners with a comprehensive but practical view of the topic (Huselid 2015). The chapter emphasizes the developments in HRODM in recent years, particularly by highlighting works that have been published within the past 5 years (Vihari and Rao 2013; Rasmussen and Ulrich 2015; Heuvel and Bondarouk 2016; Bamber et al. 2017). The second objective is to systematically analyze the literature from the ROI perspective, highlighting scientific evidence to assist decision-makers in determining how to adopt HRODM (Rasmussen and Ulrich 2015). This work aims to aid both researchers and practitioners with respect to specific directions within HRODM in which an expected ROI may be found.

Understanding what we have learned and how it has changed the ODM field helps direct future research. To this end, this chapter asks and answers three interrelated research questions (Cuozzo et al. 2017):

RQ1. What are the major themes that have been developed within HRODM research?
RQ2. What are the focus and ROI-based critique of HRODM research?
RQ3. What is the future of HRODM research?

This chapter includes four sections. The methodology section outlines the database development approach. The results and discussion sections answer the first two research questions through descriptive statistics and a critique of the results from categorizing the HRODM literature. This section also discusses how we developed and applied the ROI theoretical framework. In the third section, we answer the last research question by discussing key implications for scholars and practitioners and noting a few directions for future research. Finally, in the fourth section, we utilize two real-life examples to demonstrate HRODM implementation.

2 Methodology

The methodological approach for the review and analysis comprised four steps. First, we developed a database by undertaking a comprehensive and systematic search to identify and extract all the relevant literature on HRODM that has been published in peer-reviewed academic journals. Second, in an iterative process between theoretically derived and empirically emerging themes, we developed a template for analyzing the reviewed articles (Table 1). Third, a manual content analysis of the retrieved articles, based on the template, was used to extract descriptive and qualitative conceptual data. Finally, the results were interpreted and the findings meaningfully synthesized (Short 2009; Webster and Watson 2002). This method was used to ensure a comprehensive, meaningful, and high-quality data compilation (Cuozzo et al. 2017).

Table 1 Classification system for analyzing HRODM articles

Code	Cluster/category	Number of articles	%	Example references
E	*Empirical*	14	17	Aral et al. (2012), Bondarouk and Ruël (2013), Kandogan et al. (2014), Harrison and Getz (2015), Hou (2015), and Ramamurthy et al. (2015)
E1	Quantitative	4	30	
E2	Qualitative	6	40	
E3	Mixed methods	4	30	
C	*Conceptual*	36	45	Davenport et al. (2010), Kapoor (2010), Wiblen et al. (2010), Harris et al. (2011), Snell (2011), Minbaeva and Collings (2013), and Pape (2016)
C1	Management tools	10	28	
C2	General	18	50	
C3	Specific	8	22	
CB	*Case based*	11	14	Davenport (2006), Fitz-Enz (2000), Briggs (2011), Mondore et al. (2011), Boyd and Gessner (2013), Singh and Roushan (2013), Varshney et al. (2014), Frigo et al. (2015), and Russell and Bennett (2015)
CB1	General	4	36	
CB2	Specific	7	64	
T	*Technical*	19	24	Karasek (2015), Kazakovs et al. (2015), Korpela (2015), Momin and Mishra (2015), Perrin (2015), Stone et al. (2015), Ulrich and Dulebohn (2015), Welbourne (2015), and Ryan and Herleman (2016)
T1	Informative	9	47	
T2	Specific	5	26	
T3	Literature review	3	16	
T4	HRA trends	2	11	

Based on: Chalutz Ben-Gal (2019)

2.1 Database Development

The initial step comprised the identification of the relevant research. To capture previously published research, we used 11 EBSCO online databases.[1] We conducted a Boolean search using "human resources analytics" as a key search term within the

[1] Databases included for the review: Business Source Premier; EconLit; Regional Business News; SocINDEX; ERIC; Library, Information Science & Technology Abstracts; Historical Abstracts; Communication & Mass Media Complete; GreenFILE; Political Science Complete; PsycARTICLES.

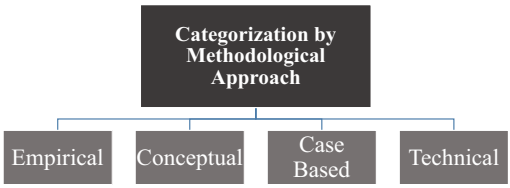


Fig. 1 Human resources organizational data-mining (HRODM) clusters. (Based on: Chalutz Ben-Gal (2019))

title, abstract or subject terms.[2] We continuously updated the database throughout the period of our research project by means of a Google Scholar alert specific to our key terms. The selection criteria are based on the following items: (1) the paper was published between 2000 and 2016, (2) the search terms appear in the title, abstract, or paper, and (3) the paper appears in a peer-reviewed journal. Overall, the searches resulted in a database of 80 articles.

2.2 Categorization

In reviewing and analyzing the selected papers, we identified four HRODM research clusters: empirical, conceptual, case-based, and technical. These research clusters are depicted in Fig. 1. This categorization is useful in developing an ROI-based analysis of HRODM (Webster and Watson 2002; Gilbert et al. 2008; Bukhari et al. 2017).

2.3 Classification System

The articles were first coded by the lead category (i.e., cluster) and then checked for consistency by an external judge who had extensive experience with the topic. Any discrepancies were reviewed and discussed before a final classification was agreed upon. Rather than describe each category in the framework as presented here in Table 1, we outline each at the beginning of the corresponding discussion in the descriptive results (Cuozzo et al. 2017).

3 Results and Discussion

In this section, we use descriptive statistics and commentary to answer the first two research questions: RQ1. What are the major themes that have been developed within HRODM? RQ2. What are the focus and ROI-based critique of HRODM? The

[2] Additional search terms included "organizational data mining," "workforce analytics," "people analytics," and "human capital analytics."

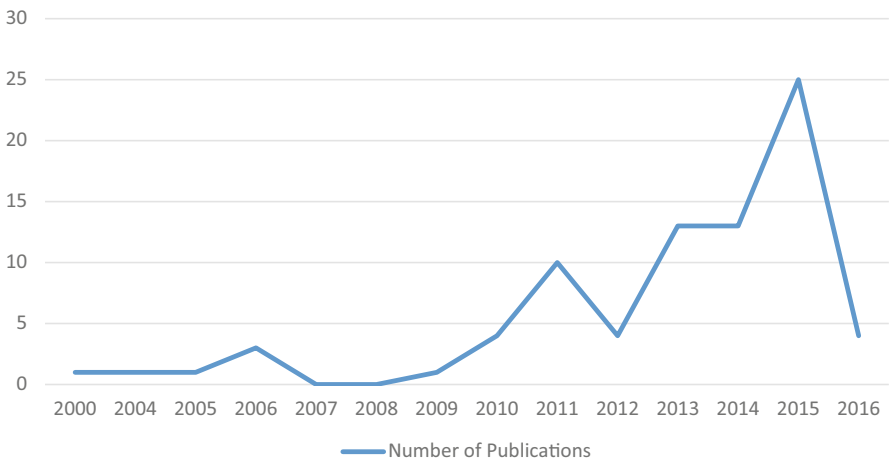


Fig. 2 HRODM publications over time. (Based on: Chalutz Ben-Gal (2019))

data reported in Fig. 2 and in Tables 1, 2, 3, and 4 form the basis for this section. Additionally, the discussion is complemented by further analysis that delves deeper than the descriptive results.

We analyze the findings of our systematic review of a sample of 80 articles associated with research in HRODM according to the chronological development of this research (presented in Table 2 and Fig. 2). We thereby identify shifting trends over time and extract key themes of existing HRODM literature. Additionally, we analyze and present key trends in HRODM research in Table 3 in a unique synthesis (presented below).

3.1 Emergence of HRODM Research

The results of our research, displayed in Table 2, clearly show an increasing interest in the topic of HRODM over time (see also Fig. 2). We identified three periods of HRODM research. The first is a period of incubation (2000–2005) during which 4% of the HRODM research was published. The second was a period of incremental growth (2006–2010) during which 10% of the HRODM research was published. Finally, there was a period of substantial growth (2011–2016) when 86% of the HRODM research was published. In line with this typology, and consistent with previous research (Rasmussen and Ulrich 2015; Nemati and Barko 2002, 2003), our study results demonstrate that the research attention devoted to HRODM has increased in recent years. The shift in publication over this 17-year period underscores the growing academic interest in the field of HRODM (Bose 2015; Kazakovs et al. 2015). Moreover, the understanding of HRODM has changed over time. While early publications examined HRODM from a narrow economic

Table 2 HRODM research characteristics by period of publication

Year of publication	2000–2005 (6 years) Incubation period*(1 ≥ publications)N* = 3 (4%)	2006–2010 (5 years) Incremental growth*(1 < publications < 4)N* = 8 (10%)	2011–2016 (6 years) Substantial growth*(publications ≥ 10)N* = 69 (86%)	2000–2016 Total (17 years)*N* = 80 (100%)
Type of journal				
HRM	2(3)	4(5)	32(40)	38(48)
Management & Business	0	4(5)	30(37)	34(43)
Engineering	0	0	2(3)	2(3)
Other	1(1)	0	5(6)	6(7)[a]
Research cluster				
Empirical	2(3)	1(1)	11(14)	14(18)
Conceptual	1(1)	3(4)	32(40)	36(45)
Case Based	0	2(3)	9(11)	11(14)
Technical	0	2(3)	17(21)	19(24)[a]
Geographical region				
North America	1(1)	3(4)	49(61)	53(66)
Europe	2(3)	2(3)	8(10)	12(15)
Asia	0	2(3)	8(10)	10(13)
Africa/Middle East	0	1(1)	4(5)	5(6)

Based on: Chalutz Ben-Gal (2019)
Values = Number of articles; values in brackets = % of articles
[a]Adds to 101% due to percentage rounding

Table 3 Trends in HRODM research

Trend	Challenges & outcomes	ROI	Example references
HRODM as strategic management tool	Management-HR-DM interface Business impact	High	Levenson (2005, 2015), Newcomer and Brass (2015), Welbourne (2015), and Xiu et al. (2017)
Evidence-based approach in HRODM	Adoption of correct tool Technological	High	Bassi (2011), Nemati and Barko (2002, 2003), McIver et al. (2018), and Strohmeier (2018)
HRODM as decision-making support tool	Various analytical techniques Multi-step process	High	Pessach et al. (2020), Dulebohn and Johnson (2013), Singh and Roushan (2013), Holsapple et al. (2014), Rasmussen and Ulrich (2015), Pape (2016), and Chamorro-Premuzic et al. (2017)
HRODM as management fad	HRODM is not part of DM HR professional's role in HRODM	Low	Rasmussen and Ulrich (2015) and Nemati and Barko (2003)

Based on: Chalutz Ben-Gal (2019)

perspective by highlighting technical aspects (e.g., Lazear 2000), the relevance of HRODM has gained importance both in research and in practice from a strategic and managerial perspective, which has transformed it into a vibrant and interesting topic of research.

More specifically, HRODM research has evolved such that in the incubation period (2000–2005), none of the publications found their way into management nor business journals, whereas almost 40% (37%) of the publications did so in the substantial growth period (2011–2016). Moreover, the study results indicate that a vast share of HRODM research (91%) was published in either HR management or in management and business journals. Forty-eight percent of HRODM research was published in HR management journals, while 43% of HRODM research was published in management and business journals.

These findings indicate the increase in the strategic importance of the field. One explanation is the growing centrality of human capital as a key organizational asset (Bontis and Fitz-Enz 2002; Fitz-Enz 2000; Nemati and Barko 2002, 2003). Both HR and ODM as a broader field are in a constant state of change (Bamber et al. 2017; McIver et al. 2018). A second explanation is the growing availability of readily accessible data, which can be transformed into valuable and actionable insights through the implementation of ODM tools (Macan et al. 2012; Strohmeier 2018). These findings also show that our ROI-based analysis is an appropriate platform to expand upon in order to determine precisely how management HRODM.

Research results suggest an emerging shift over time regarding the geographical regions upon which HRODM research focuses. Most articles on the topic of HRODM that specified a geographical region in the substantial growth period shifted from Europe (10% of publications) to North America (61% of publications).

Table 4 ROI-based[a] analysis of HRODM

Study Authors	Research Cluster	Logic	Analytics	Measurements	Processes	ROI
Harrison and Getz	Empirical	×		×	×	High
Hou	Empirical	×	×	×	×	High
Ramamurthy et al.	Empirical	×	×	×	×	High
Sharif	Empirical	×		×	×	High
Bose	Conceptual	×			×	Medium
Church et al.	Conceptual	×			×	Medium
Huselid	Conceptual	×		×	×	High
Levenson	Conceptual	×	×	×	×	High
Momin and Mishra	Conceptual	×			×	Medium
Newcomer and Brass	Conceptual	×			×	Medium
Perrin	Conceptual	×			×	Medium
Rasmussen and Ulrich	Conceptual	×			×	Medium
Sharma et al.	Conceptual	×			×	Medium
Steffi et al.	Conceptual	×				Low
Ulrich and Dulebohn	Conceptual	×			×	Medium
Zang and Ye	Conceptual	×			×	Medium
Zhao and Carlton	Conceptual	×	×	×	×	High
Frigo et al.	Case Based	×				Low
Russell and Bennett	Case Based	×		×		Medium
Chong and Shi	Technical	×				Low
Karasek	Technical	×				Low
Kazakovs et al.	Technical				×	Low
Korpela	Technical	×				Low
Stone et al.	Technical	×			×	Medium
Welbourne	Technical	×				Low

Based on: Chalutz Ben-Gal (2019)
$N = 25$. Included in 2015 publications analysis only (represents the highest publishing year in the Substantial Growth period)
[a]Boudreau and Ramstad (2006)

This focus on North America could be linked to the emerging trend, which originated in the United States, of linking technology and data along with the major effect that technology has on organizations as a whole (Chamorro-Premuzic et al. 2017).

Most of the research articles are conceptual (45%) rather than purely technical (24%). The conceptual studies in HRODM provide management and analytical tools to facilitate working processes and procedures. They include talent analytics (Burdon and Harpur 2014), tools for improved organizational decision-making (Minbaeva and Collings 2013; Pape 2016), and a conceptual framework (Boudreau and Ramstad 2006). The focus has thereby shifted over time from a predominance of conceptual articles to technical articles, which comprise nearly one-quarter (24%) of the total number of articles (see Table 2). To a certain extent, this may be due to

the growing interest in specific topics within ODM (Macan et al. 2012; Yadav 2014; Momin and Mishra 2015).

3.2 Trends in HRODM Research

Our integrative review reveals that HRODM research is dominated by four trends (see Table 3). We synthesize these trends by depicting their key challenges, outcomes, and ROI.

The first identified trend in HRODM research is the exploration of HRODM as a strategic management tool. This approach yields a high ROI for the organization because its impact may be on the organization as a whole and on the business level for the purpose of continuous improvement (Delbridge and Barton 2002). Where HRODM is presumed to be an integral part of management processes, the key challenges associated with this trend include answering questions regarding specific strategic measures. One example is organizational key performance indicators (KPIs), for example: turnover and churn, which have a long-term business impact on the organization as a whole (Levenson 2005, 2015; Newcomer and Brass 2015; Welbourne 2015). Along these lines, one researched theme associated with this trend is the management and organizational interfaces within organizations (Huselid 2015; Xiu et al. 2017; McIver et al. 2018).

The second identified trend in HRODM research is the evidence-based approach to organizational data mining research. This approach also yields a high ROI for the organization because it uses a variety of methodological and technological tools to predict improved individual or organizational performance. The key challenges associated with this trend include answering key questions regarding which tool would be the correct one to adopt for a specific people analytic challenge and which form of technology to use (Strohmeier 2018).

The third identified trend in HRODM research that uses ODM for effective organizational processes involves incorporating data mining as an effective decision-making support tool (Dulebohn and Johnson 2013; Singh and Roushan 2013; Holsapple et al. 2014; Chamorro-Premuzic et al. 2017). The ROI associated with this trend is high because it suggests efficiency in the decision-making processes. The key challenges associated with this trend include the efficiency of the process itself, e.g., collecting and analyzing the data, thereby raising issues of efficiency and effectiveness (Rasmussen and Ulrich 2015; Pape 2016; Dastyar et al. 2017).

Finally, the studies focusing on the future of ODM incorporate a fourth trend in HRODM research. This approach yields a low ROI because it is speculative in nature. The key challenges associated with this trend include discussions of whether HRODM should be part of the HR function and the role of HR professionals (Rasmussen and Ulrich 2015). This paper builds upon this trend and pinpoints specific practical directions regarding how to implement HRODM. We thus move to

our substantive contribution, an ROI-based analysis of HRODM that sets the ground for our proposed future research avenues in ODM.

3.3 Theoretical Framework: ROI-Based Analysis of HRODM

The theoretical framework of ROI guided our analysis. The literature suggests that ROI is an important measurement tool that may assist stakeholders in managerial decision-making. ROI is rooted in early theoretical research in the accounting and management professions that aimed to provide a qualitative approach to decision-making. ROI is also used in various academic fields (Philips 2012; Bontis and Fitz-Enz 2002; Bukhari et al. 2017). One example is in the corporate training and education literature, where ROI is used to measure the impact of training and educational investments on an organization's "bottom line," i.e., organizational performance measures (Charlton and Osterweil 2005).

We examine the results of this study from an ROI-based perspective for two reasons. First, we believe that this framework is suitable in light of the limited high-quality research that has been conducted in the field (Fink 2010; Handa and Garima 2014; Xiu et al. 2017). Second, we believe that analyzing HRODM from an ROI-based perspective can increase the chances of the practical adoption of HRODM. We therefore categorized the research reported in this article based on the LAMP framework (Boudreau and Ramstad 2006). We identified this framework as a suitable framework to analyze ROI in the field of HRODM. In particular, the LAMP framework assists in analyzing useful components of HRODM, i.e., "logic," "analysis," "measurement," and "process" (Boudreau and Ramstad, p. 27). Using this categorization, we found that the majority of articles from the empirical and conceptual research clusters resulted in high or medium levels of ROI. Additionally, we found that most studies focusing on cases or technical aspects of HRODM resulted in medium or low levels of ROI. We summarize the coding of our sample in Table 4.

3.4 Empirical Studies

Empirical studies attempt to obtain knowledge in the field of HRODM. The majority of studies were conducted by direct and indirect observations and/or experience (Aral et al. 2012). Their analyses were either quantitative or qualitative. An advantage of empirical studies is that by quantifying evidence or making sense of it in a qualitative manner, scholars answer empirical questions that are clearly defined and answerable based on data and the use of the evidence collected. The research designs vary by field and by the question being investigated. Some scholars perform mixed-methods research, combining qualitative and quantitative forms of analysis to better answer their research questions, especially in the social sciences

and education (Gilbert et al. 2008; Aral et al. 2012; Kandogan et al. 2014; Fire and Puzis 2016).

The contributions of empirical studies in the literature are evident as they explore new and current trends in HRODM research in all or some of the following ways. First, they conduct interviews with practitioners in a variety of organizations from different industries on the topic of HRODM. Additionally, they conduct interviews with thought leaders in the area of human capital analytics and research. Finally, they attempt to draw informative conclusions in the area of HRODM (Lazear 2000; Fink 2010; Hausknecht 2014; Kandogan et al. 2014; Sharif 2015).

As documented in Table 4, a review of the literature suggests that most empirical studies apply the LAMP model (Boudreau and Ramstad 2006) in a meaningful manner, thereby yielding a high level of ROI. Moreover, our results indicate that most empirical studies are consistent with the LAMP model because they focus on at least three of the model's components by providing meaningful content to the "logic," "analytics," "measurement," or "process" of HRODM (Boudreau and Ramstad 2006; Gilbert et al. 2008). Moreover, empirical studies yield the highest ROI because they focus on organizational practices and business performance. Furthermore, empirical studies highlight the metrics used by organizations as well as the impact of HRODM on business outcomes (Lazear 2000; Lawler III et al. 2004). Finally, empirical studies tend to use strategic tools, such as forecasting techniques to predict various human-related measures (Hausknecht 2014; Bondarouk and Ruël 2013; Del Angizan et al. 2014).

To conclude, there appear to be clear benefits to exploring HRODM from an empirical standpoint. Some of the benefits include increased organizational performance, greater accuracy regarding performance specifications, accurate and rapid assessment processes, and better HR processes (Harrison and Getz 2015; Hou 2015; Ramamurthy et al. 2015).

3.5 Conceptual Studies

Some of the studies covered in this systematic review offer conceptual contributions to the field of HRODM. The advantage of the conceptual studies is that their contributions are wide; they range from providing management tools (Davenport et al. 2010; Wiblen et al. 2010; Kapoor 2010; Snell 2011; Harris et al. 2011) to providing an ethical perspective to talent analytics (Burdon and Harpur 2014) and adopting a data-mining-based approach (Ramamurthy et al. 2015). Their contributions relate to various content areas in the HRODM field (Gilbert et al. 2008). Some studies apply statistics, technology, and expertise to large sets of people data, which results in improved organizational decisions (Minbaeva and Collings 2013; Pape 2016). Other studies emphasize analytical processes to enhance an organization's competitive advantage (Burdon and Harpur 2014).

Reviewing the conceptual literature, we identified four main themes: Organizational data mining's ROI, the conceptual framework contribution, the Orga-

nizational data mining process, and the domain of Organizational data mining. The conceptual contribution of Organizational data mining's ROI is discussed by Levenson (2005), Ingham (2011), Huselid (2015), Rasmussen and Ulrich (2015), and Zang and Ye (2015). A conceptual framework contribution is provided by Boudreau and Ramstad (2006). A discussion of the ODM process is provided by Baron (2011), Hota and Ghosh (2013), Dulebohn and Johnson (2013), Handa and Garima (2014), and Bose (2015). Finally, the domain of organizational data mining is presented and discussed by Carlson and Kavanagh (2011).

Interestingly, the results presented in Table 4 indicate that like the empirical studies, the majority of conceptual studies apply the LAMP model (Boudreau and Ramstad 2006) in a meaningful manner, thereby yielding a medium to high level of ROI. Moreover, our results indicate that most conceptual studies are consistent with the LAMP model because they focus on at least two of the model's components, i.e., "logic," "analytics," "measurement," or "process" (Boudreau and Ramstad 2006).

Conceptual studies in HRODM yield a medium to high ROI because some propose new frameworks to analyze and implement organizational and employee data (Davenport et al. 2010; Wiblen et al. 2010; Garcea et al. 2011), while others discuss the roles and responsibilities of HR in this transformational era of technological change and globalization (Kapoor 2011; Snell 2011; Harris et al. 2011; Burdon and Harpur 2014). Some of the reviewed literature focuses on performance management (Schläfke et al. 2012; Ding and Zhang 2014; Church et al. 2015; Ryan and Herleman 2016)[3] and may provide a new method for managers to obtain insight into the effectiveness of employee performance and, ultimately, organizational performance (Ding and Zhang, p. 5). Some of the conceptual studies take a broader approach to the measurement of human capital in light of constant organizational change (Baron 2011; Carlson and Kavanagh 2011; Ingham 2011; Dulebohn and Johnson 2013).

The increased level of ROI that is derived from the conceptual literature on HRODM (medium to high level of ROI, as indicated in Table 4) is also derived from some of its strategic implications. Some conceptual studies in HRODM provide tools for workforce analytics and emphasize the strategic importance of HRODM within the organizational context (Huselid and Becker 2011; Van Barneveld et al. 2012; Hota and Ghosh 2013; Boudreau 2014; Guszcza and Richardson 2014; Handa and Garima 2014; Holsapple et al. 2014; Bose 2015; Huselid 2015; Rasmussen and Ulrich 2015; Ryan and Herleman 2016; Sharma et al. 2015; Steffi et al. 2015; Zang and Ye 2015; Ulrich 2016).

To conclude, the common feature of the conceptual studies is that they articulate a clear connection between the investment in analytics and organizational effectiveness. Moreover, they all have indicators of increased level of ROI. Finally, the conceptual research studies present a robust approach for strategic alignment

[3] The literature on performance management analytics focuses on business, sales and individual performance. This review includes the last.

with state-of-the-art organizational processes (Boudreau and Ramstad 2006), which complements their overall effectiveness.

3.6 Case-Based Studies

The case-based literature has two foci. First, it covers studies that provide practical examples of organizations that have implemented HRODM and recommendations for successful implementation. Second, some studies were written by scholars or practitioners who have consulting experience in the area of HRODM and share it with their readers. An advantage of the case-based studies is their practicality in the field of HRODM (Gilbert et al. 2008).

The results presented in Table 4 indicate that in contrast to the empirical and conceptual studies, most case-based studies do not apply the LAMP model (Boudreau and Ramstad 2006) in a meaningful manner, and they therefore yield a medium to low level of ROI. Moreover, our results indicate that most case-based studies are inconsistent with the LAMP model and therefore yield lower levels of ROI because they focus on only one or two of the model's components, i.e., "logic," "analytics," "measurement," or "process" (Davenport 2006; Fitz-Enz 2000; Briggs 2011; Mondore et al. 2011; Boyd and Gessner 2013; Singh and Roushan 2013; Varshney et al. 2014; Frigo et al. 2015; Russell and Bennett 2015).

To conclude, the common grounds for what we categorized as case-based studies (Gilbert et al. 2008) is that the majority do not articulate a clear connection between HRODM investment, organizational effectiveness, and ROI. Moreover, they provide limited scientific evidence to aid decision-makers concerning whether to adopt or implement organizational data mining tools within an organization (Strohmeier 2018).

3.7 Technical Studies

The technical literature analyzed in this study has four focus areas: The studies present informative research on the topic of HRODM (Welbourne 2015), focus on a specific subject within HRODM (Perrin 2015), present a literature review (Vihari and Rao 2013; Chong and Shi 2015), or illustrate future trends in HRODM (Yadav 2014; Momin and Mishra 2015). Thus, the advantage of technical studies is their specificity (Gilbert et al. 2008).

The results presented in Table 4 indicate that similar to the case-based literature, and in contrast to the empirical and conceptual studies, the majority of technical studies do not apply the LAMP model (Boudreau and Ramstad 2006) in a meaningful manner, and they therefore yield a medium to low level of ROI. Moreover, our results indicate that most technical studies are inconsistent with the LAMP model because they focus on only one or two of the model's components, i.e., "logic,"

"analytics," "measurement," or "process" (Mayo 2006; Rivera and Smolders 2013; Stone and Dulebohn 2013; Vihari and Rao 2013; Fernández-Delgado et al. 2014; Yadav 2014; Chong and Shi 2015; Karasek 2015; Kazakovs et al. 2015; Korpela 2015; Momin and Mishra 2015; Perrin 2015; Stone et al. 2015; Ulrich and Dulebohn 2015; Welbourne 2015; Ryan and Herleman 2016).

To conclude, the common ground of what we categorized as technical studies (Gilbert et al. 2008) is that similar to case-based studies, most papers do not articulate a clear connection between HRODM investment and organizational effectiveness. Moreover, they provide limited scientific evidence to aid decision-makers concerning whether to adopt organizational data mining.

4 HRODM: Practical Implementation Tools and Expected ROI

4.1 Implications for Organizations

Our review of the literature underscores the importance of two notable fields within the HRODM research, namely, empirical and conceptual research. We further explore specific HRODM tasks and challenges in light of practical implementation tools and the expected ROI within organizational functions (Bassi 2011; Buede et al. 2018).

From a practical perspective, the ROI-based approach presented is important for a data-driven decision-making process in the field of HRODM. It also provides a step-by-step procedure for handling data and subsequently utilizing these data to attain meaningful managerial insights. Moreover, the need for a better focus in conducting and implementing HRODM projects within organizations is clear. Albeit with an element of shortage, some HRODM efforts in organizations today could be defined as reactive rather than proactive. Hence, it is not unusual for practitioners to use data that they receive access to in order to perform interesting analyses by addressing a question or set of questions with various levels of viability to the organization (Huselid 2015). The ROI-based approach to HRODM presented in this study provides a robust tool to compare and contrast different dilemmas and associated values that can be derived from conducting various types of ODM projects. The ROI-based approach also supports continuous improvement in organizations (Delbridge and Barton 2002).

From a theoretical perspective, the proposed categorization (Gilbert et al. 2008) provides a robust ROI framework for conducting research in the field of HRODM, thus enabling scholars and practitioners to focus on a desired topic in a more structured manner (Becker 2009; Lipkin 2015; Rasmussen and Ulrich 2015; Ghosh and Sengupta 2016; Pape 2016).

In Table 5, we illustrate the implications of HRODM for organizations. We present how addressing organizational challenges using various analytical tools,

Table 5 HRODM implications for organizations: Practical implementation tools & expected ROI

Task	Sample challenges	Tool[a]	Expected ROI
Industry Analysis	Macro market effect on turnover	Descriptive	Low
Workforce Planning	High-demand jobs and attrition Person-Organization Fit	Predictive	High
Job Analysis	Robustness of job components	Descriptive	Low
Recruitment and Selection	Person-Job Fit	Predictive	High
Training and Development	ROI in training	Descriptive and Predictive	Medium
Compensation	Total compensation scenarios	Descriptive and Predictive	Medium
Performance Management	Performance management cycle scenarios	Descriptive	Low
Retention	Can retention be predicted	Descriptive and Predictive	Medium

Based on: Chalutz Ben-Gal (2019)

[a]*Descriptive analytics* tools may include: Descriptive statistics, Graphs and plots, Benchmarking tools, KPIs-Based Methods (scorecards), Business Intelligence (BI) Dashboards, and Advanced Survey Analytics

Predictive analytics tools may include: Market Basket Analysis, Regression and parametric modeling (including Logistic Regression), Time Series Analysis, Classification Methods (e.g. decision trees, SVM, Discriminant Analysis, Neural Networks, Deep Learning), Clustering (K nearest neighbors, K-means,) Anomaly detection, Profiling, Association rules, Link-Analysis, Causality modeling (Bayesian networks), Text Analysis & NLP and Attrition Modelling

namely, descriptive and predictive, may impact the expected ROI. This analysis may further assist scholars and practitioners in the ongoing effort to improve HRODM tools and impacts (Rasmussen and Ulrich 2015; Chamorro-Premuzic et al. 2017; Strohmeier 2018).

Table 5 presents the implications of HRODM for organizations as well as practical implementation tools. Specifically, it offers a summary overview of eight major tasks and activities that organizations are faced with, including their corresponding sample challenges (Srinivasan and Chandwani 2014; Bamber et al. 2017), practical implementation tools, and the expected ROI. The expected ROI is categorized into three levels – low, medium, and high – in accordance with the complexity of data-handling procedures that are relevant to the HRODM research (Fitz-Enz 2009; Rasmussen and Ulrich 2015; Ghosh and Sengupta 2016). The results documented in Table 5 yield two notable conclusions. The two areas of tasks that yield the highest ROI are workforce planning and recruitment and selection because both rely on predictive analytics tools (Fitz-Enz 2009).

As presented in Table 5, the first task focuses on *industry analysis*. This task ensures the analysis of basic parameters in an organization's specific industry (e.g., retail, financial, technology). Empirical research tools are descriptive analytics that use BI and benchmarking to analyze government data, consulting firms' data, census data, and macro-industry data. Our observations suggest examining macro market

effects on specific constructs, such as turnover. Relevant ratios include industry average job turnover, average cost per hire, and job-specific retention budget, among others. Accordingly, the ROI for performing an industry analysis utilizing these HRODM tools is expected to be low.

Workforce planning is the second task, and we call for extended empirical analytics on this task because we believe it entails a high ROI. Workforce planning ensures the use of a continual process to align the needs and priorities of the organization with those of its workforce to ensure that it can meet its legislative, regulatory, service and production requirements as well as long- and short-term organizational objectives (Huselid 2015). Empirical research tools include predictive analytics that use various analysis techniques based mostly on internal data (e.g., ERP, headcount, product mapping, financials, budget) and external data (e.g., surveys, salary tables, syllabuses, and training program materials). Our observations suggest that certain challenges to test are person-organization fit and the connection between high-demand jobs and attrition. Accordingly, the ROI for performing workforce planning using these HRODM tools is expected to be high.

The third task focuses on *job analysis*, which is a process to identify and determine in detail a given job's duties, requirements, and interfaces as well as its relative importance. This is a process in which judgments are made about data collected for a job (Levenson 2005). Empirical research tools include descriptive analytical tools (e.g., financial ERP, organizational structure, and headcount). Our observations suggest that some specific challenges to test are the robustness of job components and their effect on satisfaction and retention. Accordingly, the ROI for performing job analysis using these HRODM tools is expected to be low.

Recruitment and selection of talent is the fourth task, and we call for extended empirical research on this task because we believe it entails a high ROI. Practical research tools include predictive analytics. Our observations suggest that specific challenges to test are methods of classifying the talent pool according to available organizational resources; text analysis of interviews and profiling of vacant roles and organizational requirements; and logistics regression or other parametric models that predict recruitment probability of success, satisfaction, and person-job fit. Accordingly, the ROI for the recruitment and selection of talent using these HRODM tools is expected to be high.

The fifth task refers to *training and development*, which is primarily concerned with organizational activity aimed at improving the performance of individuals and groups in the organization. The recommended empirical research tools include both descriptive and predictive analytics. Our observations suggest that some specific challenges to test are the analysis of training ROI (through BI), whereas classification methods may assist in improving the training investment per job class. Accordingly, the ROI for performing training and development using these HRODM tools is expected to be at a medium level.

The sixth task refers to *compensation and benefits*. This management challenge assists in the execution of organizational strategy and may be adjusted according to business needs, goals, and available resources. Empirical research tools are descriptive (e.g., BI, scorecards, or other KPI-based methods) and predictive analytics.

Our observations suggest that some specific challenges are total compensation scenario testing; Monte Carlo simulations assess various compensation plans and regression analyses and their interplay with selected organizational phenomena. Accordingly, the ROI for performing compensation research using these HRODM tools is expected to be at a medium level.

The seventh task refers to *performance management*. This task is an ongoing process of communication between a supervisor and an employee that occurs throughout the year in support of accomplishing the organization's strategic objectives (Huselid 2015). Future empirical research tools are based on descriptive analytics. Our observations suggest that some specific challenges to explore are performance management cycle scenarios mainly through BI, dashboarding, and KPI-based methods. Additionally, various levels of performance are clustered for the purpose of performance evaluation. Accordingly, the ROI for performing performance management using these HRODM tools is expected to be low (Buede et al. 2018).

Finally, the eighth task that is illustrated in Table 5 is *retention of talent*. The recommended empirical research tools are based on descriptive and predictive analytics. We call for a specific challenge to test and believe that further research on the topic of whether retention can be predicted is required. This challenge can be addressed by the profiling of key jobs, the classification of various talent retention scenarios, logistic regression, anomaly detection, and attrition modeling for various job groups. Accordingly, the ROI of performing talent retention using the proposed HRODM tools is expected to be at a medium level.

Our observations documented above apply to both scholars and practitioners when planning their future HRODM priorities. Furthermore, the ROI-based approach, which is the focus of this study, underscores the call for a more systematic approach for researchers and decision-makers to use evidence-based information as a guide to the adoption of HRODM and to understand its effectiveness (Rasmussen and Ulrich 2015; Buede et al. 2018).

The challenge for future research in HRODM is to reach beyond general studies in order to identify important contextual variables of HRODM and to consistently add value to existing organizational systems on both the contextual and practical levels. As emphasized earlier, we believe that much of this potential added value lies within the empirical and conceptual research in HRODM. Therefore, fertile avenues for future research contributions should focus on both empirical and conceptual studies in HRODM since these are the noticeable directions where the highest return rates are expected. Enhancing and developing empirical and conceptual knowledge in HRODM and state-of-the-art tools may serve as adequate future contributions to the field of HRODM.

If decision-makers have ROI information to guide the adoption of HRODM, a more focused and systematic research approach must evolve. Macro-organizational theoretical frameworks can add to the ROI-based approach by proposing different perspectives. For example, the contextual approach (Johns 2006, 2018) may offer a basis for understanding the organizational context in which specific ROI is to be found in line with new scholarly insights in the HRODM field. Additionally,

further theory development may integrate the LAMP framework (Boudreau and Ramstad 2006) with contextual elements (Johns 2006, 2018), which may also offer an appealing framework for testable hypotheses. Future rigorously constructed research questions may focus on the various tasks of HR from a holistic point of view while challenging the recommended analytical approach presented in this paper. Finally, future research may propose a new methodology that differs from the ROI-based approach to systematically analyze the scholarly and practical field of HRODM.

5 Contributions: ROI – Model to Guide the Way Forward

This section answers the third research question, RQ3: What is the future of HRODM research? It also reiterates the study's contributions and emphasizes the ROI approach as a model to guide the way forward in HRODM research.

HRODM is a fascinating, dynamic discipline (Levenson 2011; Huselid 2015). The dynamic role of ODM enables it to focus both on the operational tasks of HR and on the organizations' long-term and strategic business objectives. The growing field of HRODM enables management and engineering scholars and executives to implement a broader approach, which may increase their strategic contribution (Kazakovs et al. 2015; Strohmeier 2018). Machine-learning and data analytics in general, and more specifically in the field of HRODM, can aid in making informed decisions based on knowledge extracted from the available data and options (Sharma et al. 2015).

Our unique synthesis of the literature underscores the importance of two important fields within HRODM, namely, the empirical and the conceptual research. Our observations that we analyze and discuss in this study offer an ROI-based perspective to the HRODM field. Moreover, the ROI-based approach on the topic of HRODM presented in this paper provides theoretical and practical contributions. As a result, it provides a model to guide the way forward in HRODM research. From a theoretical perspective, this paper assists data analytics scholars who may find the ROI-based framework useful when fine-tuning their theoretical contributions in the field. From a practical perspective, this paper clarifies the dilemma associated with the HRODM field and assists practitioners regarding the expected ROI of HRODM initiatives within their organizations.

In conducting our ROI-based review of the literature on HRODM by integrating the analysis above, several major conclusions emerge. First, there is a need for more scientific empirical research in HRODM. Focusing on the development of such research might increase the potential for action-oriented, data-driven research, which can assist management and technical professionals. Second, as with the previous conclusion, and in light of notable deficiencies in the existing HRODM literature (Boudreau and Ramstad 2006; Dastyar et al. 2017), there emerges a need to focus on an ROI-based approach, which is our proposed model to guide the way forward.

We have taken a step toward systematically explaining some notable questions in the HRODM field. Not only does a focus on an ROI-based approach improve the adoption of HRODM as an important field in data science, but the context in which it is being adopted and implemented also matters, both practically and theoretically speaking.

6 HRODM Implementation: Two Examples

In this section, we provide two examples of HRODM implementation. The first example presents a *Workforce Analytics and Big Data Analysis of Employee Turnover* example – taken from Sela and Chalutz Ben-Gal (2018). The second example presents an *Employee Recruitment Decision-Making Support Tool* taken from Pessach et al. (2020).

6.1 HRODM Implementation Example I: Workforce Analytics and Big Data Analysis of Employee Turnover

Workforce analytics and turnover are strategic domains in HRODM implementation (Chalutz Ben-Gal 2019; Hausknecht 2014). This example shows that by utilizing HRODM tools for the purpose of workforce analytics through big data analysis and cluster analysis, management can gain a nuanced perspective on employee turnover and career path trajectories. This knowledge is important for strategic workforce planning across various industries. Furthermore, the digital world we live in results in weaker social connections between individuals. This drives the development of new workforce models, which are based on social networks and artificial intelligence (Fire and Puzis 2016; Aghabaghery et al. 2020). For example, in recent years, labor markets experience fundamental changes (e.g., social networks platforms and artificial intelligence-based recruitment and placement processes). Additionally, more candidates utilize online-based tools in their job search (Sela and Chalutz Ben-Gal 2018).

In this numerical example, taken from Sela and Chalutz Ben-Gal (2018), the authors extracted a unique dataset of over 970,000 curriculum vitas (CVs) based on LinkedIn profiles. Their analysis revealed just a slightly opposite relation between employee job satisfaction and turnover rate. The researchers found that while higher compensation packages provided by companies often lead to higher employee job satisfaction, they do not ensure lower turnover rates (Hausknecht 2014). The authors demonstrate that by utilizing HRODM tools, surprising and non-intuitive patterns are revealed, especially in global high-technology companies (see Fig. 3a, b).

In their research, the authors implemented the following methodology. They calculated the average employment period based on a dataset of 973,134 CV's

retrieved from active LinkedIn profiles. Their dataset included 44 features, including control variables. For example, gender, country of employment, seniority, endorsed skills, as well as employment archival data – company name, job title, employment duration in previous firms, industry sector, etc. The dataset was merged with two additional benchmark data sources (Fortune 100; Glassdoor). The authors present their analysis and findings in Fig. 3.

Figure 3a presents machine-learning-based big data analysis of employee turnover. Figure 3a presents the average employee job satisfaction level, the average

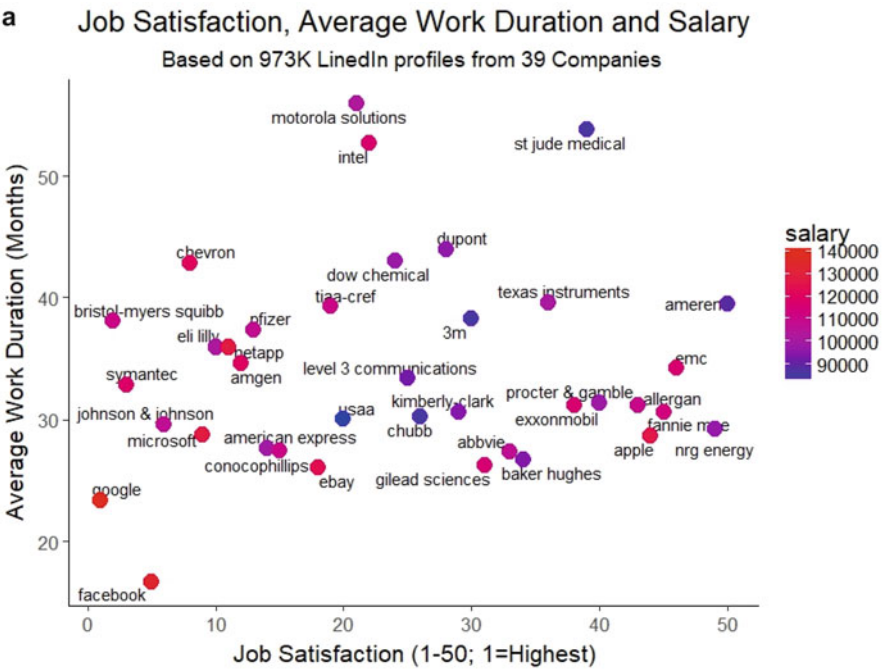


Fig. 3 (**a**–**d**) HRODM implementation: Example I – workforce analytics and big data analysis of employee turnover. (**a**) Big data analysis of employee turnover. Average employee job satisfaction (x-axis), average work duration in months (y-axis) and average salary in 39 organizations. Organizations such as Facebook or Google, have high salaries, high employee job satisfaction level, however a low average employment duration. In comparison, Intel has an average level of employee job satisfaction, but longer working durations. Symantec and Bristol-Myers Squibb both offer lower salary levels, relatively higher employee job satisfaction levels, and longer employment durations. Finally, Apple and EMC have rather low employee job satisfaction levels, high-average salary levels, and average work durations. (**b**) Analysis of employee turnover. Comparison of employment (work) duration histograms for eight companies: Facebook, Google, eBay, IBM, Apple, 3M, Intel, and Motorola (ordered by descending average employment periods). (**c**) Workforce Analytics of Career Paths. Three career path network clusters: financial cluster (red); consulting cluster (green); and the high technology cluster (blue). (**d**) Workforce analytics of career paths. Analysis of career paths within and across career network clusters reveals that Facebook and Google dominate two distinct employment clusters; IBM is a "Career Hub". (Taken from: Sela and Chalutz Ben-Gal (2018))

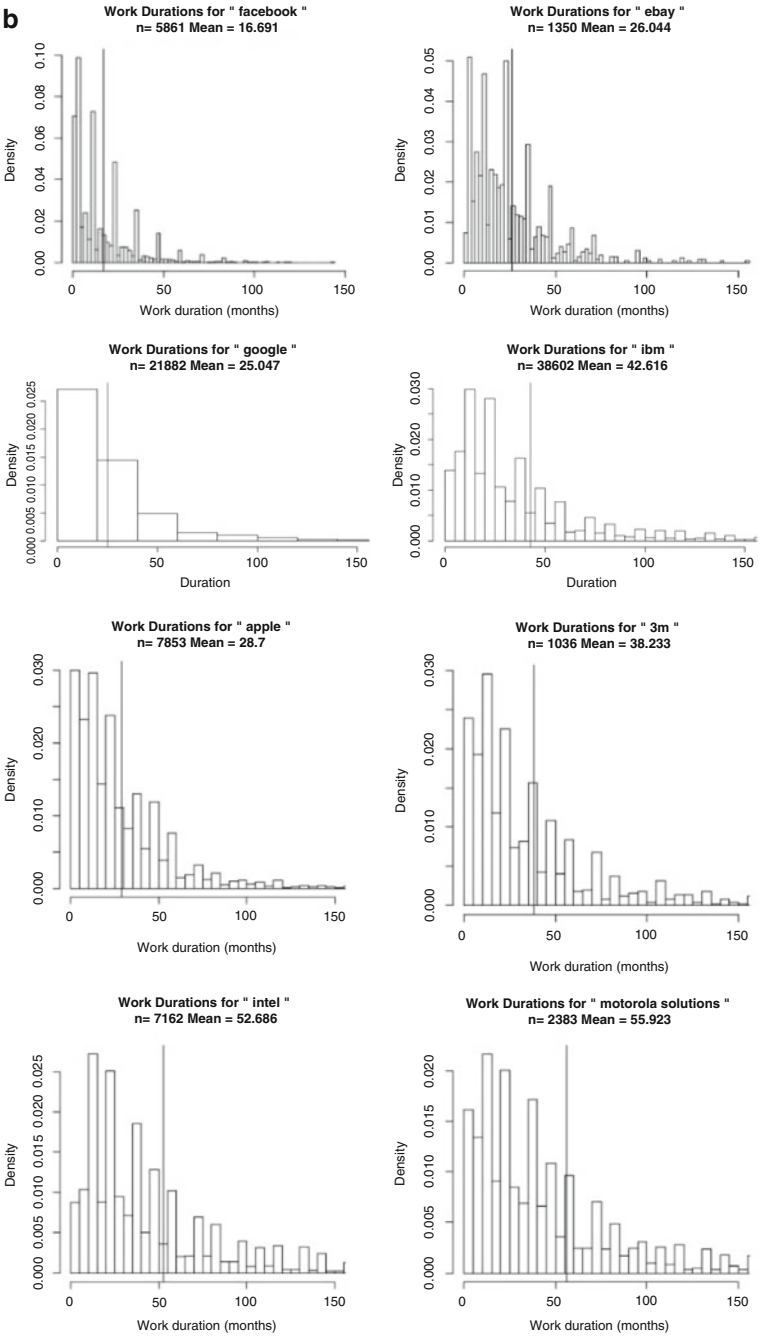


Fig. 3 (continued)

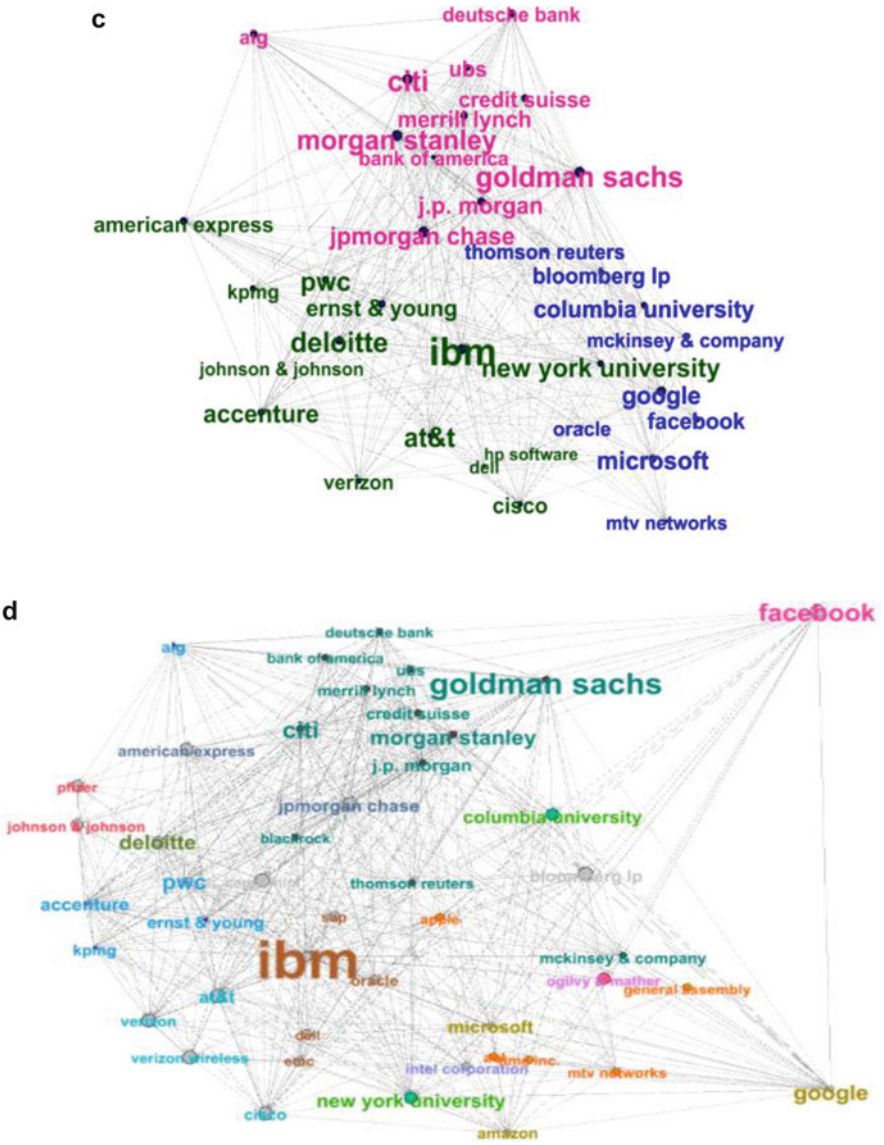


Fig. 3 (continued)

employment (work) duration, and the average salary in thirty-nine organizations. As demonstrated by the authors in Fig. 3a, the companies Google and Facebook are located in the bottom left corner. These companies offer higher compensation packages (the red color indicates a compensation package of 130–140 k USD per year), and demonstrate higher job satisfaction levels. Utilizing HRODM tools,

the authors are able to demonstrate some surprising findings. For example, the researcher's results indicate that Facebook has an average employment duration of 16.9 months. As extracted from their research and illustrated in Fig. 3a, organizations such as Facebook or Google, which offer high compensation packages, which seem to influence higher employee job satisfaction levels, demonstrate a surprisingly low employment periods for their employees.

The researchers compare and contrast their findings to other organizations. For example, they found that Intel, which demonstrates an average employee job satisfaction score, shows a relatively very long employment duration. They also found that, both Symantec and Bristol-Myers Squibb have lower compensation packages, however demonstrate higher job satisfaction levels, as well as longer employment durations. Finally, the researchers found that Apple or EMC have low job satisfaction scores, high compensation packages, and demonstrate average employment durations.

HRODM tools enable, researchers and practitioners alike, further data-driven analysis of the workforce. Furthermore, HRODM tools enable a granular view of the data. In Fig. 3b Sela and Chalutz Ben-Gal (2018) present a comparison of eight company's employment (work) duration histograms. In Fig. 3b, the authors present histograms representing employment duration for eight technology companies. For example, while the upper three figures (representing Facebook, eBay, and Google) resemble an exponential function, the lower two histograms (representing Intel and Motorola) have a peak at an employment period of approximately 24 months, and smaller bins in lower and in higher values. The authors conclude that assuming an entry-level job, which usually requires about 6 months up to 1 year in order to reach mastery level of performance, these patterns are unexpected compared to Facebook or Google, where the average employment period is 16.9 and 23.3 months, respectively.

The authors emphasize the counterintuitive nature of their findings and claim that as Fig. 3a, b indicate, despite higher levels of employee job satisfaction and higher compensation packages, both Facebook and Google demonstrate shorter employment periods.

However, it is management's role to analyze, interpret, and ultimately act upon such results, thus maximizing organizational ROI (Chalutz Ben-Gal 2019). Therefore, HRODM tools may assist the understanding of such macro market phenomena. For example, technology companies such as Google, Facebook, and eBay may be perceived as technological trendsetters, thus serve as "career platforms" for candidates toward their next desired job or professional challenge (Sela and Chalutz Ben-Gal 2018).

HRODM tools enable a deeper understanding of the workforce flows by analyzing employees' career trajectory patterns. The researchers further illustrate an HRODM implementation example across industries. In Fig. 3c, the authors present three main career network clusters extracted from the LinkedIn dataset: the financial cluster (represented in red color), the consulting cluster (represented in green color), and the high technology cluster (represented in blue color).

The cluster analysis methodology presented by the authors, a popular tool within the HRODM domain, enables to detect employment and career moves for the purpose of workforce analytics. The authors claim that examining Fig. 3c, it seems that employees tend to make more frequent career moves within their own career network cluster. For example, employees working in the financial cluster (represented in red color), tend to make career moves to other financial companies, but less frequently to other career network clusters. Similar career trajectory patterns exist in the consulting cluster (represented in green color), and the high technology cluster (represented in blue color). Moreover, within the consulting cluster, the authors identified IBM as a "career hub," which they define as a company that serves as a central crossroad junction for employees from which they can easily transfer to a different career network cluster. The author's finding is surprising when compared to their Facebook and Google results, because both companies' positioning represent a less central point as potential employers, and thus do not serve as industry hubs. Consequently, the authors conclude that one can detect that working at IBM may serve as a strategic career bridge to other attractive employment industries, compared to working in other companies, in which employees are more likely to stay within the borders of their career network cluster, especially when performing career choices (Sela and Chalutz Ben-Gal 2018).

Additional findings are illustrated by the authors in Fig. 3d, in which it is shown that both Facebook and Google are companies that dominate two distinct employment clusters (i.e., industries). Applying HRODM tools, the researchers analyzed employment clusters within a network that consists of almost 50 thousand individual career moves. They found that only one single career move was performed between these two companies. They also found that IBM is centrally located within its employment cluster, based on its physical central location, as well as on its evident large size.

This example emphasizes HRODM utility in the integration of machine-learning and data-driven tools, in order to maximize organizational ROI (Sela and Chalutz Ben-Gal 2018; Chalutz Ben-Gal 2019; Aghabaghery et al. 2020).

6.2 HRODM Implementation Example II: Employee Recruitment Decision-Making Support Tool

Employee recruitment is a strategic domain for HRODM implementation and is associated with high organizational ROI (Chalutz Ben-Gal 2019; Pessach et al. 2020), because it improves fit levels (Johns 2006, 2018) between a candidate and a specific position to be staffed. Recruitment and selection of talent is an organizational task associated with predictive analytics tools. Some HRODM application domains include methods of the talent pool classification; text analysis of interviews and profiling of vacant jobs compared to organizational demand;

prediction models for recruitment probability of success. Accordingly, the ROI for the recruitment and selection of talent using HRODM tools is expected to be high (Vihari and Rao 2013; Chalutz Ben-Gal 2019).

In this numerical example, taken from Pessach et al. (2020), the authors extracted a unique dataset and illustrated an application of a decision-making support tool for organizations and for the Human Resources community in order to improve the accuracy of recruitment and placement decisions. The example utilizes HRODM-driven machine-learning models for the prediction of recruitment success, as well as for extracting interpretable insights.

The authors measure the recruitment success based on a combination of the candidate turnover rate (Hausknecht 2014; Sela and Chalutz Ben-Gal 2018), and an objective target indicator based on performance. The authors also measure the performance indicator based on the position-changing conditions. For the purpose of classification and prediction of successful and unsuccessful recruitments and placements, as well as for mining significant patterns, the researchers use a Variable-Order Bayesian Networks Model (VOBN) (Ben-Gal et al. 2005; Singer and Ben-Gal 2007).

The authors evaluate the model compared to other machine-learning models applied to an extracted unique organizational dataset. The dataset utilized by the researchers includes about 700,000 cases of employee candidates who were recruited to an organization throughout a period of a decade (hired between the years 2000 and 2010). The authors detail some pre-hire features in the dataset. For example, position requirement, age, gender, marital status, education, grades, skills, interview and test scores, professional preferences, and additional socio-demographic features. Furthermore, the authors describe pre-processing activities. For example, data tables consolidation, sensitive data masking, etc.

In line with HRODM tools and techniques, the authors identified several clusters of position groups. Furthermore, using statistical data extracted from the Central Bureau of Statistics enabled the researchers to enrich the dataset with additional socio-demographic features. Missing values were tagged by zeros, and candidates with many missing values were removed by the researchers. In line with HRODM practices, additional dimensionality reduction procedures were performed by the researchers in accordance with the applied machine-learning algorithms.

The authors classified a "successful" and "unsuccessful" recruitment process as follows: They analyzed key reasons for employee turnover and categorized them into two groups: "successful recruitment" group, i.e., turnover associated with "natural" reasons, such as promotion (Hausknecht 2014), and an "unsuccessful recruitment" group, i.e., unexpected turnover, such as short-term or poor performance based turnover. Additionally, turnover was classified by the researchers as negative (e.g., "misfit") or positive (e.g. "promotion"). Finally, the combination of turnover and position changes was utilized as a combined measure for labeling "successful" vs. "unsuccessful" recruitment.[4]

[4] In line with HRODM practices, and in order to maintain consistency, the researchers applied an a-priori distribution of the target class on both a training and a testing datasets.

In this example, the authors were able to prove that the VOBN Model performs well in terms of both interpretability and accuracy in predicting recruitment success because it identifies context-based patterns that can support the organization in the recruitment process. Therefore, the authors explain that it can be used to extract rules and actions for the recruiters who sometimes lack the HRODM and machine-learning background, providing actionable and implementable insights – See Fig. 4a, b below. This HRODM implementation example is clear and easy to understand, therefore allows for an examination HR policies and procedures by what the authors call "extraction of interpretable and actionable insights" (Pessach et al. 2020).

In Fig. 4a – taken from Pessach et al. (2020) – the authors present the predicted probabilities of success in assigning sixteen candidates of two types to four positions by the machine-learning models (e.g., VOBN Model). For clarity purpose, the authors present colored entries in order to differentiate between the success probability values (i.e., high probability marked in green and low probability marked in red).

a

Candidate type	Candidate ID	Position 1409	Position 1509	Position 379	Position 40
type 1	Candidate 7317	0.67	0.68	0.68	0.48
	Candidate 8320	0.47	0.68	0.48	0.56
	Candidate 9346	0.53	0.68	0.39	0.56
	Candidate 3145	0.61	0.68	0.51	0.55
	Candidate 5438	0.63	0.68	0.48	0.57
	Candidate 0142	0.67	0.68	0.51	0.51
	Candidate 1617	0.55	0.68	0.45	0.55
type 2	Candidate 8610	0.66	0.68	0.85	0.58
	Candidate 2939	0.48	0.68	0.85	0.60
	Candidate 4349	0.56	0.73	0.85	0.60
	Candidate 4842	0.69	0.68	0.85	0.55
	Candidate 7983	0.69	0.58	0.86	0.59
	Candidate 6405	0.64	0.68	0.86	0.59
	Candidate 7882	0.59	0.62	0.86	0.48
	Candidate 5481	0.84	0.68	0.85	0.54
	Candidate 0226	0.62	0.69	0.83	0.54

Fig. 4 (**a**, **b**) HRODM implementation: Example II – employee recruitment decision-making support tool. (**a**) Predicted probabilities of success in assigning sixteen candidates of two types of populations to four positions. The entries are color-coded by the success probability values. High probability (green), low probability (red). (**b**) Assignment of candidates to positions by four different solutions. Solution 1 (red) suggests the following: (i) recruiting 4 candidates to position 1409; (ii) recruiting 6 candidates to position 1509; (iii) recruiting 6 candidates to position 379 (note that none of them are of type 1); and (iv) not recruiting any of the candidates to position 40. (Taken from: Pessach et al. (2020))

b

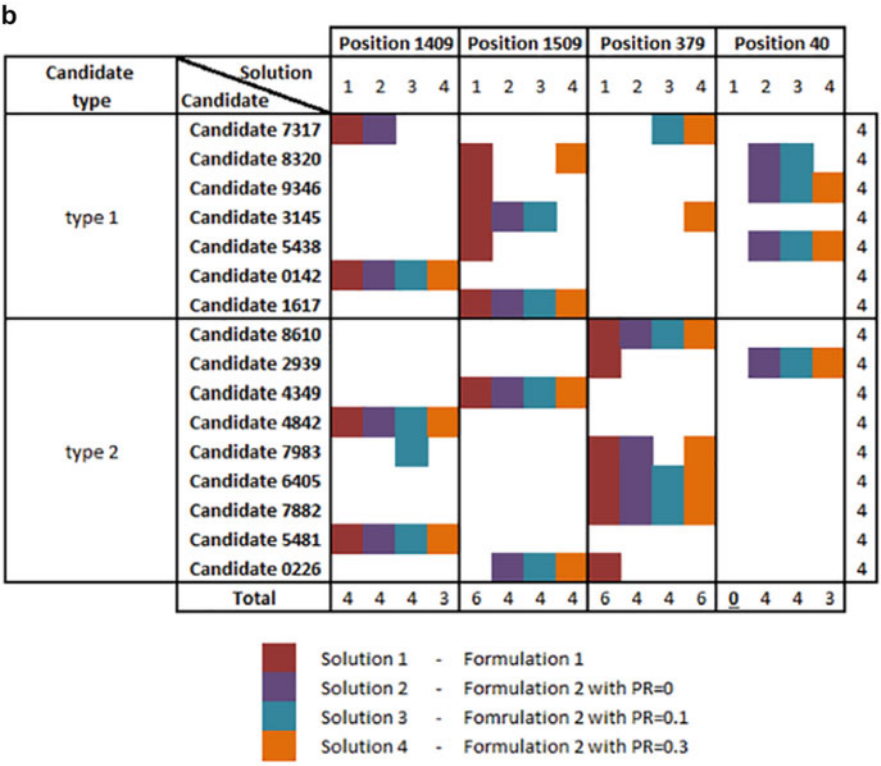


Fig. 4 (continued)

The data presented by the researchers in Fig. 4a includes four positions (columns), sixteen candidates (rows), and two types of candidates who need to be assigned in a pre-determined way (e.g., based on their specific background, skills, and departmental demand). Figure 4a also presents the predicted success probability for each pair of candidate and position (the authors use shades of green to represent high probability and shades of red to represent low probabilities). For calculation purpose, the authors assumed a demand of six employees per position. Analyzing their dataset under these constraints, the authors conclude that based on the machine-learning algorithms if the goal is to maximize the sum of recruitment success probabilities, then position 379 requires staffing by type 2 candidates only.

In Fig. 4b, the researchers present four solutions to the allocation problem based on four different formulations where the rows represent candidates and the columns represent positions.

This example illustrates how HRODM techniques can be implemented as an employee recruitment decision-making support tool. This tool can support managers and recruiters alike when seeking candidates to be placed in target vacant positions. Moreover, the proposed decision-making tool illustrated in this example can further

assist the HR function in making relevant strategic decision. For example, employee development and retention procedures, in order to maximize organizational ROI (Chalutz Ben-Gal 2019; Pessach et al. 2020).

References

Aghabaghery, R., Golpayegani, A. H., & Esmaeili, L. (2020). A New Method for Organizational Process Model Discovery Through the Analysis of Workflows and Data Exchange Networks. *Social Network Analysis and Mining*, 10(1), 12.

Aral, S., Brynjolfsson, E., & Wu, L. (2012), "Three-Way Complementarities: Performance Pay, Organizational data mining and Information Technology", *Management Science*, 58, pp. 913–931.

Bamber, G. J., Bartram T., Stanton, P. (2017), "HRM and workplace innovations: formulating research questions", *Personnel Review*, 46 (7), pp. 1216–1227.

Baron, A. (2011), "Measuring human capital", *Strategic HR Review*, 10(2), pp. 30–35.

Bassi, L. (2011), "Raging debates in Organizational data mining", *People & Strategy*, 34, pp. 14–18.

Becker, G. S. (2009), "Human capital: A theoretical and empirical analysis, with special reference to education", University of Chicago press.

Ben-Gal, I., Shani, A., Gohr, A., Grau, J., Arviv, S., Shmilovici, A., & Grosse, I. (2005). Identification of transcription factor binding sites with variable-order Bayesian networks. *Bioinformatics*, 21(11), 2657–2666.

Bondarouk, T., & Ruël, H. (2013), "The strategic value of e-HRM: results from an exploratory study in a governmental organization", *The International Journal of Human Resources Management*, 24(2), pp. 391–414.

Bontis, N., & Fitz-Enz, J. (2002). Intellectual capital ROI: a causal map of human capital antecedents and consequents. *Journal of Intellectual Capital*, 3(3), pp. 223–247.

Bose, M. T. (2015), "Growing Impact of Human Capital Analytics on modern economy: A Generic overview", *Journal of Business and Management*, 17(1), pp. 11–13.

Boudreau, J. W., & Ramstad, P. M. (2006), "Talentship and HR measurement and analysis: From ROI to strategic organizational change", *People and Strategy*, 29(1), pp. 25–33.

Boudreau, J. (2014), "Will HR's grasp match its reach? An estimable profession grown complacent and outpaced", *Organizational Dynamics*, 43(3), pp. 189–197.

Boyd, N., & Gessner, B, (2013), "Human resources performance metrics: methods and processes that demonstrate you care", *Cross Cultural Management: An International Journal*, 20(2), pp. 251–273.

Briggs, H. (2011), "How to use a data-focused approach to embed good HR practices", *Strategic HR Review*, 10(2), pp. 18–23.

Buede, D. M., Axelrad, E. T., Brown, D. P., Hudson, D. W., Laskey, K. B., Sticha, P. J., & Thomas, J. L. (2018). Inference enterprise models: An approach to organizational performance improvement. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(6), e1277.

Burdon, M., & Harpur, P. D. (2014), "Re-conceptualising privacy and discrimination in an age of talent analytics", *University of New South Wales Law Journal*, 37(2), p. 679.

Bukhari, H., Andreatta, P., Goldiez, B., & Rabelo, L. (2017), "A framework for determining the return on investment of simulation-based training in health care". INQUIRY: *The Journal of Health Care Organization, Provision, and Financing*, 54, 0046958016687176.

Carlson, K. D., & Kavanagh, M. J. (2011), "HR metrics & workforce analytics", MJ Kavanagh, RD.

Chalutz Ben-Gal, H. (2019), An ROI-based review of HR analytics: Practical Implementation Tools, *Personnel Review*, Vol. 48 No. 6, pp. 1429–1448. https://doi.org/10.1108/PR-11-2017-0362.

Chamorro-Premuzic, T., Akhtar, R., Winsborough, D., & Sherman, R. A. (2017), "The datafication of talent: how technology is advancing the science of human potential at work"' *Current Opinion in Behavioral Sciences*, (18), pp. 13–16.

Charlton, K. & Osterweil, C. (2005), "Measuring return on investment in executive education: a quest to meet client needs or pursuit of the holy grail?", *The Ashridge Journal*, pp. 6–13

Chong, D., & Shi, H. (2015), "Big data analytics: a literature review", *Journal of Management Analytics*, 2(3), pp. 175–201.

Church, A. H., Ginther, N. M., Levine, R., & Rotolo, C. T. (2015), "Going Beyond the Fix: Taking Performance Management to the Next Level", *Industrial and Organizational Psychology*, 8(01), pp. 121–129.

Cuozzo, B., Dumay, J., Palmaccio, M., & Lombardi, R. (2017). Intellectual capital disclosure: a structured literature review. *Journal of Intellectual Capital*, 18(1), pp. 9–28.

Dastyar, B., Kazemnejad, H., Sereshgi, A. A., & Jabalameli, M. A. (2017). Using Data Mining Techniques to Develop Knowledge Management in Organizations: A Review. *Journal of Engineering, Project, and Production Management*, 7(2), 80.

Davenport, T. H. (2006), "Competing on analytics", *Harvard Business Review*, 84 (1), p. 98.

Davenport, T. H., Harris, J., & Shapiro, J. (2010), "Competing on talent analytics", *Harvard Business Review*, 88(10), pp. 52–58.

Del Angizan, S., Navid, B. J., & Soratiyan, G. (2014), "Modeling the impact of data integration on the functional elements of human resources management", *International Research Journal of Applied and Basic Sciences*, 8(7), pp. 926–932.

Delbridge, R., & Barton, H. (2002), "Organizing for continuous improvement: structures and roles in automotive components plants", *International Journal of Operations & Production Management*, 22(6), pp. 680–692.

Ding, C., & Zhang, J. (2014), "Diagnosis for Effectiveness: Performance Assessment from a Diagnostic Measurement Perspective with Big Data", *International Journal of Business and Social Research*, 4(6), pp. 1–11.

Dulebohn, J. H., & Johnson, R. D. (2013), "Human resources metrics and decision support: A classification framework", *Human resources Management Review*, 23(1), pp. 71–83.

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014), "Do we need hundreds of classifiers to solve real world classification problems?", *The Journal of Machine-learning Research*, 15(1), pp. 3133–3181.

Fink, A. A. (2010), "New trends in human capital research and analytics", *People and Strategy*, 33(2), p. 14.

Fire, M., & Puzis, R. (2016). Organization mining using online social networks. *Networks and Spatial Economics*, 16(2), 545–578.

Fitz-Enz, J. (2000). *The ROI of human capital*. Amacom, New York.

Fitz-Enz, J. (2009), "Predicting people: from metrics to analytics", *Employment Relations Today*, 36(3), pp. 1–11.

Fortune 100, 100 best companies to work for. http://fortune.com/best-companies/list/.

Frigo, M. L., Ubelhart, M. C., Eccles, R. G., & Youmans, T. (2015), "CFO+ CHRO= POWER PAIR", *Strategic Finance*, 97(5), p. 26.

Fulmer, I. S., & Ployhart, R. E. (2013), "Our Most Important Asset: A Multidisciplinary/Multilevel Review of Human Capital Valuation for Research and Practice", *Journal of Management*, 0149206313511271.

Garcea, N., Isherwood, S., & Linley, A. (2011), "Do strengths measure up?", *Strategic HR Review*, 10(2), pp. 5–11.

Ghosh, A., & Sengupta, T. (2016), "Predictive analytics for human resources", edited by J. Fitz-Enz and II John Mattox, Hoboken, NJ, John Wiley and Sons, 2014, pp. 1–149.

Gilbert, B. A., McDougall, P. P., & Audretsch, D. B. (2008). Clusters, knowledge spillovers and new venture performance: An empirical examination. *Journal of Business Venturing*, 23(4), pp. 405–422.

Glassdoor, Company Review API. https://www.glassdoor.com.

Guszcza, J., & Richardson, B. (2014), "Two dogmas of big data: Understanding the power of analytics for predicting human behavior", *Deloitte Review*, (15), pp. 161–175.

Handa, D. & Garima, A. (2014), "Human resources (HR) Analytics: Emerging Trends in HRM", *International Journal of Research in Commerce & Management*, 5(6), pp. 59–62.

Harris, J. G., Craig, E., & Light, D. A. (2011), "Talent and analytics: new approaches, higher ROI", *Journal of Business Strategy*, 32(6), pp. 4–13.

Harrison, J. L., & Getz, C. (2015), "Farm size and job quality: mixed-methods studies of hired farm work in California and Wisconsin", *Agriculture and Human Values*, pp. 1–18.

Hausknecht, J. P. (2014), "Collective data on collective turnover: What factors most affect turnover rates?", (CAHRS ResearchLink No. 4). Ithaca, NY: Cornell University, ILR School, Center for Advanced Human resources Studies.

Heuvel, S., & Bondarouk, T. (2016), "The rise (and fall) of Organizational data mining: a study into the future applications, value, structure, and system support", Article submitted for the 2nd HR Division International Conference (HRIC) on February 20–22, 2016 in Sidney, Australia.

Holsapple, C., Lee-Post, A., & Pakath, R. (2014), "A unified foundation for business analytics", *Decision Support Systems*, 64, pp. 130–141.

Hota, J., & Ghosh, D. (2013), "Workforce Analytics Approach: An Emerging Trend of Workforce Management", *Workforce Management*, pp. 167–179.

Hou, C. K. (2015), "Using the balanced scorecard in assessing the impact of BI system usage on organizational performance. An empirical study of Taiwan's semiconductor industry", *Information Development*, 0266666915614074.

Huselid, M. A., & Becker, B. E. (2011), "Bridging micro and macro domains: Workforce differentiation and strategic human resources management", *Journal of Management*, 37(2), pp. 421–428.

Huselid, M. A., (2015), "The Rise of HR: Wisdom from 73 Thought Leaders", HR Certification Institute Publisher.

Ingham, J. (2011), "Using a human capital scorecard as a framework for analytical discovery", *Strategic HR Review*, 10(2), pp. 24–29.

Johns, G., (2006), "The essential impact of context on organizational behavior", *Academy of Management Review*, 31 (2), pp. 386–408

Johns, G. (2018). Advances in the treatment of context in organizational research. *Annual Review of Organizational Psychology and Organizational Behavior*, 5, 21–46.

Kandogan, E., Balakrishnan, A., Haber, E. M., & Pierce, J. S. (2014), "From Data to Insight: Work Practices of Analysts in the Enterprise", Computer Graphics and Applications, *IEEE*, 34(5), pp. 42–50.

Kapoor, B. (2010), "Business Intelligence and Its Use for Human resources Management" *The Journal of Human resources and Adult Learning*, 6(2), pp. 21–30.

Kapoor, B. (2011), "Impact of globalization on human resources management", *Journal of International Management Studies*, 6(1), p. 1.

Kapoor, B., & Sherif, J. (2012), "Human resources in an enriched environment of business intelligence", *Kybernetes*, 41(10), pp. 1625–1637.

Karasek, A. (2015), "Information Technologies in Human resources Management-Selected Examples", World Academy of Science, Engineering and Technology, *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, 9(6), pp. 1837–1842.

Kazakovs, M., Verdina, A., & Arhipova, I. (2015), "Automation of Human resources Development Planning", *Procedia Computer Science*, 77, pp. 234–239.

Kharb, L. (2019). Data Mining: A Distinctive Approach to CRM. *International Journal of Scientific Research in Network Security and Communication*, 7(2), 6–10.

Korpela, K. (2015), "Improving cyber security awareness and training programs with data analytics", *Information Security Journal: A Global Perspective*, 24(1–3), pp. 72–77.
Lawler III, E. E., Levenson, A., & Boudreau, J. W. (2004), "HR metrics and analytics–uses and impacts", *Human resources Planning Journal*, 27(4), pp. 27–35.
Lazear, E. P. (2000), "The future of personnel economics", *The Economic Journal*, 110(467), pp. 611–639.
Levenson, A. (2005), "Harnessing the power of Organizational data mining", *Strategic HR Review*, 4(3), pp. 28–31.
Levenson, A. (2011), "Using targeted analytics to improve talent decisions", *People and Strategy*, 34(2), p. 34.
Levenson, A. (2015), "Strategic Analytics: Advancing Strategy Execution and Organizational Effectiveness", Berrett-Koehler Publishers.
Lipkin, J. (2015, September 4), "Sieving through the data to find the person: HR's imperative for balancing big data with people centricity", [Electronic version] Cornell HR Review. Retrieved [10.4.2016] from Cornell University, ILR School site: http://digitalcommons.ilr.cornell.edu/chrr/84
Macan, T., Lemming, M. R., & Foster, J. L. (2012), "Utility analysis: do estimates and format matter?", *Personnel Review*, 42(1), pp. 105–126.
Mayo, A. (2006), "Measuring and reporting–the fundamental requirement for data. What's the Future for Human Capital?"
McIver, D., Lengnick-Hall, M. L., & Lengnick-Hall, C. A. (2018). A strategic approach to workforce analytics: Integrating science and agility. *Business Horizons*.
Meghyasi, H., & Rad, A. (2020). Customer Churn Prediction in Irancell Company by Using Data Mining.
Minbaeva, D., & Collings, D. G. (2013), "Seven myths of global talent management", *The International Journal of Human resources Management*, 24(9), pp. 1762–1776.
Momin, W. Y. M., & Mishra, K. (2015), "Organizational data mining as a Strategic Workforce Planning", *IJAR*, 1(4), pp. 258–260.
Mondore, S., Douthitt, S., & Carson, M. (2011), "Maximizing the impact and effectiveness of Organizational data mining to drive business outcomes", *People and Strategy*, 34(2), p. 20.
Nemati, H. R., & Barko, C. D. (2002). Enhancing enterprise decisions through organizational data mining. *Journal of Computer Information Systems*, 42(4), 21–28.
Nemati, H. R., & Barko, C. D. (2003). Key factors for achieving organizational data-mining success. *Industrial Management & Data Systems*, 103(4), 282–292.
Newcomer, K., & Brass, C. T. (2015), "Forging a strategic and comprehensive approach to evaluation within public and nonprofit organizations: Integrating measurement and analytics within evaluation", *American Journal of Evaluation*, 1098214014567144.
Pape, T. (2016), "Prioritizing data items for business analytics: Framework and application to human resources", *European Journal of Operational Research*, 252(2), pp. 687–698.
Pessach, D., Singer, G., Avrahami, D., Chalutz Ben-Gal, H., Shmueli, E., Ben-Gal, I. (2020), "Employees recruitment: A prescriptive analytics approach via machine-learning and mathematical programing", *Decision Support Systems*, 113290. https://doi.org/10.1016/j.dss.2020.113290.
Perrin, B. (2015), "Bringing accountability up to date with the realities of public sector management in the 21st century", *Canadian Public Administration*, 58(1), pp. 183–203.
Philips, J. J. (2012). *Return on investment in training and performance improvement programs*. Routledge.
Ramamurthy, K. N., Singh, M., Yu, Y., Aspis, J., Iames, M., Peran, M., & Held, Q. S. (2015, October), "A talent management tool using propensity to leave analytics", In Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on (pp. 1–10). *IEEE.*
Rasmussen, T., & Ulrich, D. (2015), "Learning from practice: how Organizational data mining avoids being a management fad", *Organizational Dynamics*, 44, pp. 236–242
Rivera, R. J., & Smolders, F. (2013), "Operational Workforce Planning is Quietly Transforming HR", *Workforce Solutions Review*, April/May.

Russell, C., & Bennett, N. (2015), "Big data and talent management: Using hard data to make the soft stuff easy", *Business Horizons*, 58(3), pp. 237–242.

Ryan, J., & Herleman, H. (2016), "A Big Data Platform for Workforce Analytics. Big Data at Work", *The Data Science Revolution and Organizational Psychology*, p. 19.

Sharma, R., Anand, A., & Coltman, T. (2015), "Creating Business Value from Digital Data Streams: The Role of Organizational Interventions", *MIS Quarterly Executive*.

Schläfke, M., Silvi, R., & Möller, K. (2012), "A framework for business analytics in performance management", *International Journal of Productivity and Performance Management*, 62(1), pp. 110–122.

Sela, A., & Chalutz Ben-Gal, H. (2018), "Big Data Analysis of Employee Turnover in Global Media Companies, Google, Facebook and Others," *2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE)*, Eilat, Israel, 2018.

Sharif, Y. M. (2015), "Research Development in Human resources Analytics in Malaysia."

Short, J. (2009), "The art of writing a review article", *Journal of Management*, 35(6), pp. 1312–1317.

Singer, G., & Ben-Gal, I. (2007). The funnel experiment: The Markov-based SPC approach. *Quality and Reliability Engineering International*, 23(8), 899–913.

Singh, N. K., & Roushan, R. K. S. (2013), "Data Analytics and Decision Support in the context of ERP and beyond for giving CSIR a Competitive advantage" – two case studies from HR (Human resources) and MM (Materials Management) Modules.

Snell, A. (2011), "Developing talent intelligence to boost business performance", Strategic *HR Review*, 10(2), pp. 12–17.

Sparrow, P. (2012), "Globalising the international mobility function: The role of emerging markets, flexibility and strategic delivery models", *The International Journal of Human resources Management*, 23(12), pp. 2404–2427.

Srinivasan, V., & Chandwani, R. (2014), "HRM innovations in rapid growth contexts: the healthcare sector in India", *The International Journal of Human resources Management*, 25(10), pp. 1505–1525.

Steffi, R. S., Baranikumar, D., & Prakash, K. (2015), "Adding the shade of green to big data analytics", *International Journal of Informative & Futuristic Research*, 2(10), pp. 3835–3841.

Stone, D. L., Deadrick, D. L., Lukaszewski, K. M., & Johnson, R. (2015), "The influence of technology on the future of human resources management", *Human resources Management Review*, 25(2), pp. 216–231.

Stone, D. L., & Dulebohn, J. H. (2013), "Emerging issues in theory and research on electronic human resources management (eHRM)", *Human resources Management Review*, 23(1), pp. 1–5.

Strohmeier, S. (2018), "Smart HRM–a Delphi study on the application and consequences of the Internet of Things in Human Resource Management", *The International Journal of Human Resource Management*, 1–30.

Ulrich, D., & Dulebohn, J. H. (2015), "Are we there yet? What's next for HR?", *Human resources Management Review*, 25(2), pp. 188–204.

Ulrich, D. (2016), "HR at a crossroads", *Asia Pacific Journal of Human resources*.https://doi.org/10.1111/1744-7941.12104

Van Barneveld, A., Arnold, K. E., & Campbell, J. P. (2012), "Analytics in higher education: Establishing a common language", EDUCAUSE learning initiative, 1, pp. 1–11.

Varshney, K. R., Chenthamarakshan, V., Fancher, S. W., Wang, J., Fang, D., & Mojsilović, A. (2014, August), "Predicting employee expertise for talent management in the enterprise", In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1729–1738). ACM.

Vihari, N. S. & Rao M. K. (2013), "Analytics as a Predictor for Strategic and Sustainable Human resources Function: An Integrative Literature Review", (Doctoral dissertation, IIT), Roorkee.

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly*, pp. xiii–xxiii.

Welbourne, T. M. (2015), "Data-Driven Storytelling: The Missing Link in HR Data Analytics", *Employment Relations Today*, 41(4), pp. 27–33.

Wiblen, S., Grant, D., & Dery, K. (2010), "Transitioning to a new HRIS: The reshaping of human resources and information technology talent", *Journal of Electronic Commerce Research*, 11(4), pp. 251–267.

Xiu, L., Liang, X., Chen, Z., Xu, W. (2017) "Strategic flexibility, innovative HR practices, and firm performance: A moderated mediation model", *Personnel Review*, 46 (7), pp. 1335–1357.

Yadav, R. (2014), "Managing global HR issues in today's challenging times".

Zang, S., & Ye, M. (2015), "Human resources Management in the Era of Big Data", *Journal of Human resources and Sustainability Studies*, 3(01), pp. 41–45.

Zhao, G., & Carlton, D. (2015), "Forecast Competency Migration by a Methodology of Competency Analytics", *Open Journal of Social Sciences*, 3(11), pp. 16–22.

Algorithmic Fairness

Dana Pessach and Erez Shmueli

1 Introduction

Nowadays, an increasing number of decisions are being controlled by artificial intelligence (AI) and machine learning (ML) algorithms. The motivation for an automated learning model is clear—we expect algorithms to perform better than human beings for several reasons: First, algorithms may integrate much more data than a human may grasp and take many more considerations into account. Second, algorithms can perform complex computations much faster than human beings. Third, human decisions are subjective, and they often include biases.

Hence, it is a common belief that using an automated algorithm makes decisions more objective or fair. However, this is unfortunately not the case since ML algorithms are not always as objective as we would expect. The idea that ML algorithms are free from biases is wrong since the assumption that the data injected into the models are unbiased is wrong. More specifically, a prediction model may actually be inherently biased since it learns and preserves historical biases (Kleinberg, Mullainathan, & Raghavan, 2017).

Since many automated decisions (for example, which individuals will receive jobs, loans, medication, bail, or parole) can significantly impact people's lives, there is great importance in assessing and improving the ethics of the decisions made by these automated systems. Indeed, in recent years, the concern for algorithm fairness has made headlines. One of the most common examples was in the field of criminal justice, where recent revelations have shown that an algorithm used by the United States criminal justice system had falsely predicted future criminality among African–Americans at twice the rate as it predicted for White people

D. Pessach (✉) · E. Shmueli
Department of Industrial Engineering, Tel Aviv University, Tel Aviv-Yafo, Israel
e-mail: danapessach@gmail.com; shmueli@tau.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_37

(Angwin, 2016). In another case of a hiring application, it was recently exposed that Amazon discovered that their ML hiring system was discriminating against female candidates, particularly for software development and technical positions. One suspected reason for this is that most recorded historical data were for male software developers (Dastin, 2018). In a different scenario in advertising, it was shown that Google's ad-targeting algorithm had proposed higher-paying executive jobs more for men than for women (Datta, Tschantz, & Datta, 2015; Simonite, 2015).

These lines of evidence and concerns about algorithmic fairness have led to growing interest in the literature on defining, evaluating, and improving fairness in ML algorithms (see, for example, Berk, Heidari, Jabbari, Kearns, & Roth, 2018; Chouldechova & Roth, 2018; Friedler et al., 2019; Holstein, Wortman Vaughan, Daumé III, Dudik, & Wallach, 2019). It is important to note, however, that the task of improving fairness of ML algorithms is not trivial since there exists an inherent trade-off between accuracy and fairness. That is, as we pursue a higher degree of fairness, we may compromise accuracy (see, for example, Kleinberg et al., 2017).

The rest of this chapter is structured as follows: Sect. 2 discusses the potential causes of algorithmic unfairness; Sect. 3 presents definitions and measures of fairness and their trade-offs; Sect. 4 reviews fairness mechanisms and methods and a comparison of the mechanisms, focusing on the pros and cons of each mechanism; Sect. 5 outlines commonly used fairness-related datasets; and Sect. 6 provides concluding remarks and sketches several open challenges for future research.

2 Potential Causes of Unfairness

The literature has indicated several causes that may lead to unfairness in machine learning (Chouldechova & Roth, 2018; Martínez-Plumed, Ferri, Nieves, & Hernández-Orallo, 2019):

- Biases already included in the datasets used for learning, which are based on biased device measurements, historically biased human decisions, erroneous reports, or other reasons. Machine learning algorithms are essentially designed to replicate these biases.
- Biases caused by missing data, such as missing values or sample/selection biases, which result in datasets that are not representative of the target population.
- Biases that stem from algorithmic objectives, which aim at minimizing overall aggregated prediction errors and therefore benefit majority groups over minorities.
- Biases caused by "proxy" attributes for sensitive attributes. Sensitive attributes differentiate privileged and unprivileged groups, such as race, gender, and age, and are typically not legitimate for use in decision-making. Proxy attributes are non-sensitive attributes that can be exploited to derive sensitive attributes. In the case that the dataset contains proxy attributes, the machine learning algorithm

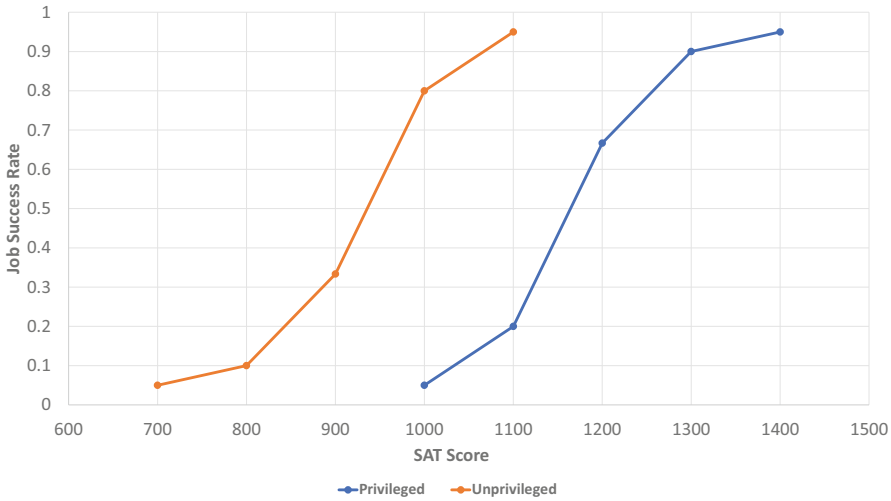


Fig. 1 If the SAT scores were used for hiring, then unprivileged candidates with high potential would be excluded, whereas lower potential candidates from the privileged group would be hired instead

can implicitly make decisions based on the sensitive attributes under the cover of using presumably legitimate attributes (Barocas & Selbst, 2016).

To illustrate the last cause mentioned above, consider the example depicted in Fig. 1. The figure illustrates a case of SAT[1] scores for two sub-populations: a privileged one and an unprivileged one.

In this illustration, SAT scores may be used to predict the probability of job success when hiring candidates since the higher the SAT score is, the higher the probability of success. However, unprivileged candidates with SAT scores of approximately 1100 perform just as well as privileged candidates with SAT scores of 1400 since they may have encountered more challenging pathways to achieve their scores. In other words, if the SAT scores were used for hiring, unprivileged candidates with high potential would be excluded, whereas lower potential candidates from the privileged group would be hired instead.

3 Fairness Definitions and Measures

This section presents some general legal notions for discrimination followed by a survey of the most common measures for algorithmic fairness, and the inevitable trade-offs between them.

[1] The SAT is a standardized test widely used for college admissions in the United States.

3.1 Definitions of Discrimination in Legal Domains

The legal domain has introduced two main definitions of discrimination: (i) **disparate treatment** (Zimmer, 1995; Barocas & Selbst, 2016): intentionally treating an individual differently based on his/her membership in a protected class (*direct discrimination*); (ii) **disparate impact** (Rutherglen, 1987; Barocas & Selbst, 2016): negatively affecting members of a protected class more than others even if by a seemingly neutral policy (*indirect discrimination*).

Put in our context, it is important to note that algorithms trained with data that do not include sensitive attributes (i.e., attributes that explicitly identify the protected and unprotected groups) are unlikely to produce *disparate treatment* but may still induce unintentional discrimination in the form of *disparate impact* (Kleinberg et al., 2017).

3.2 Measures of Algorithmic Bias

This section presents the most prominent measures of algorithmic fairness:

1. **Disparate impact** (Feldman, Friedler, Moeller, Scheidegger, & Venkatasubramanian, 2015)—This measure was designed to mathematically represent the legal notion of *disparate impact*. It requires a high ratio between the positive prediction rates of both groups. This ensures that the proportion of positive predictions is similar across groups. For example, if a positive prediction represents acceptance for a job, the condition requires the proportion of accepted applicants to be similar across groups. Formally, this measure is computed as follows:

$$\frac{P[\hat{Y} = 1|S \neq 1]}{P[\hat{Y} = 1|S = 1]} \geq 1 - \varepsilon, \tag{1}$$

   where S represents the protected attribute (e.g., race or gender), $S = 1$ is the privileged group, and $S \neq 1$ is the unprivileged group. $\hat{Y} = 1$ means that the prediction is positive. Let us note that if $\hat{Y} = 1$ represents acceptance (e.g., for a job), then the condition requires the acceptance rates to be similar across groups. A higher value of this measure represents more similar rates across groups and therefore more fairness. Note that this notion relates to the "80% rule" in disparate impact law (Feldman et al., 2015), which requires that the acceptance rate for any race, sex, or ethnic group be at least 80% of the rate for the group with the highest rate.
   Note that this notion may be applied as a measure if the right-hand side is omitted, or as a constraint if it is kept. In the case of utilizing the notion as a constraint, the value of ε might be a non-trivial task and should be determined according to the specific characteristics of each application.

2. **Demographic parity**—This measure is similar to *disparate impact*, but the difference is taken instead of the ratio (Calders & Verwer, 2010; Dwork, Hardt, Pitassi, Reingold, & Zemel, 2012). This measure is also commonly referred to as *statistical parity*. Formally, this measure is computed as follows:

$$\left| P[\hat{Y} = 1|S = 1] - P[\hat{Y} = 1|S \neq 1] \right| \leq \varepsilon. \tag{2}$$

A lower value of this measure indicates more similar acceptance rates and therefore better fairness. *Demographic parity* (and *disparate impact*) ensures that the positive prediction is assigned to the two groups at a similar rate.

One disadvantage of these two measures is that a fully accurate classifier may be considered unfair, when the base rates (i.e., the proportion of actual positive outcomes) of the various groups are significantly different. Moreover, in order to satisfy *demographic parity*, two similar individuals may be treated differently since they belong to two different groups—such treatment is prohibited by law in some cases (note that this notion also corresponds to the practice of *affirmative action* Fullinwider, 2018).

3. **Equalized odds**—This measure was designed by Hardt, Price, and Srebro (2016) to overcome the disadvantages of measures such as *disparate impact* and *demographic parity*. The measure computes the difference between the false positive rates (FPR), and the difference between the true positive rates (TPR) of the two groups. Formally, this measure is computed as follows:

$$\left| P[\hat{Y} = 1|S = 1, Y = 0] - P[\hat{Y} = 1|S \neq 1, Y = 0] \right| \leq \varepsilon \tag{3}$$

$$\left| P[\hat{Y} = 1|S = 1, Y = 1] - P[\hat{Y} = 1|S \neq 1, Y = 1] \right| \leq \varepsilon, \tag{4}$$

where the upper formula requires the absolute difference in the FPR of the two groups to be bounded by ε, and the lower formula requires the absolute difference in the TPR of the two groups to be bounded by ε. Smaller differences between groups indicate better fairness. In contrast to *demographic parity* and *disparate impact* measures, a fully accurate classifier will necessarily satisfy the two *equalized odds* constraints. Nevertheless, since *equalized odds* relies on the actual ground truth (i.e., Y), it assumes that the base rates of the two groups are representative and were not obtained in a biased manner.

One use case that demonstrates the effectiveness of this measure investigated the COMPAS (*Practitioners Guide to COMPAS*, 2012) algorithm used in the United States criminal justice system. For predicting recidivism, although its accuracy was similar for both groups (African–Americans and Caucasians), it was discovered that the *odds* were different. It was discovered that the system had falsely predicted future criminality (FPR) among African–Americans at twice the rate predicted for White people (Angwin, 2016); importantly, the

algorithm also induced the opposite error, significantly underestimating future crimes among Caucasians (FNR).

4. **Equal opportunity**—This requires true positive rates (TPRs) to be similar across groups (meaning the probability of an individual with a positive outcome to have a positive prediction) (Hardt et al., 2016). This measure is similar to equalized odds but focuses on the true positive rates only. This measure is mathematically formulated as follows:

$$\left| P[\hat{Y} = 1 | S \neq 1, Y = 1] - P[\hat{Y} = 1 | S = 1, Y = 1] \right| \leq \varepsilon, \tag{5}$$

Let us note that following the equality in terms of only one type of error (e.g., true positives) will increase the disparity in terms of the other errors (Pleiss, Raghavan, Wu, Kleinberg, & Weinberger, 2017). Moreover, according to Corbett-Davies and Goel (2018), this measure may be problematic when base rates differ between groups.

Thus far, we have mapped the most common *group* notions of fairness, which require parity of some statistical measure across groups. The literature has additionally indicated *individual* notions of fairness. It is alternatively possible to match other measures such as accuracy, error rates, or calibration values between groups (see for example Verma & Rubin, 2018). *Group* definitions of fairness, such as *demographic parity, disparate impact, equalized odds*, and *equalized opportunity*, consider fairness with respect to the whole group, as opposed to *individual* notions of fairness.

5. **Individual fairness**—This requires that similar individuals will be treated similarly. Similarity may be defined with respect to a particular task (Dwork et al., 2012; Joseph, Kearns, Morgenstern, & Roth, 2016). Individual fairness may be described as follows:

$$\left| P(\hat{Y}^{(i)} = y | X^{(i)}, S^{(i)}) - P(\hat{Y}^{(j)} = y | X^{(j)}, S^{(j)}) \right| \leq \varepsilon; \ if \ d(i, j) \approx 0, \tag{6}$$

where i and j denote two individuals, $S^{(\cdot)}$ refers to the individuals' sensitive attributes, and $X^{(\cdot)}$ refers to their associated features. $d(i, j)$ is a distance metric between individuals that can be defined depending on the domain such that similarity is measured according to an intended task. This measure considers other individual attributes for defining fairness, rather than just the sensitive attributes. However, note that in order to define similarity between individuals, a similarity metric needs to be defined, which is not trivial. This measure, in addition to assuming a similarity metric, also requires some assumptions regarding the relationship between features and labels (see, for example, Chouldechova & Roth, 2018).

3.3 Trade-Offs

Determining the right measure to be used must take into account the proper legal, ethical, and social context. As demonstrated above, different measures exhibit different advantages and disadvantages. Next, we highlight the main trade-offs that exist between different notions of fairness, and the inherent trade-off between fairness and accuracy.

Fairness Measures Trade-Offs

Interestingly, several recent studies have shown that it is not possible to satisfy multiple notions of fairness simultaneously (Berk et al., 2018; Chouldechova, 2017; Corbett-Davies & Goel, 2018; Corbett-Davies, Pierson, Feller, Goel, & Huq, 2017; Friedler, Scheidegger, & Venkatasubramanian, 2016; Kleinberg et al., 2017; Pleiss et al., 2017). For example, when base rates differ between groups, it is not possible to have a classifier that equalizes both calibration and odds (except for trivial cases such as a classifier that assigns all examples to a single class). Additionally, there is also evidence for the incompatibility between equalized accuracy and equalized odds, as in the COMPAS criminal justice use case (Angwin, 2016; Berk et al., 2018).

Pleiss et al. (2017) recommend that in light of the inherent incompatibility between equalized calibration and equalized odds, practical implications require choosing only one of these goals according to the specific application's requirements. We recommend that any selected measure of algorithmic fairness be considered in the appropriate legal, social, and ethical contexts.

Fairness–Accuracy Trade-Off

The literature extensively discusses the inherent trade-off between accuracy and fairness—as we pursue a higher degree of fairness, we may compromise accuracy (see for example Kleinberg et al., 2017). A theoretical analysis of the trade-off between fairness and accuracy was studied in Corbett-Davies et al. (2017) and Lipton, Chouldechova, & McAuley (2017). Since then, many papers have empirically supported the existence of this trade-off (for example, Friedler et al., 2019; Menon & Williamson, 2018; Bechavod & Ligett, 2017a). Generally, the aspiration of a fairness-aware algorithm is to achieve a model that allows for higher fairness without significantly compromising the accuracy or other alternative notions of utility.

It is worth noting that in this chapter, we refer to fairness that relates to human beings; however, the term "fair" is also used in other applications in a different sense. For example, in concurrent computing systems, an algorithm is considered fair, if it ensures an equitable resource allocation, such that no process is starving (starvation happens when a process is denied adequate resources that are required for its execution) (Mutlu & Moscibroda, 2008; Pandey & Shanker, 2018). As seen in this section, we refer to notions of fairness that are different from the above-mentioned concurrent computing fairness.

For further reading about algorithmic fairness measures, we refer the reader to Corbett-Davies and Goel (2018), Kleinberg et al. (2017), and Verma and Rubin (2018).

4 Fairness-Enhancing Mechanisms

Numerous recent papers have proposed mechanisms to enhance fairness in machine learning algorithms. These mechanisms are typically categorized into three types: pre-process, in-process, and post-process. The following three subsections review studies in each one of these categories. The fourth subsection is devoted to comparing the three mechanism types and providing guidelines on when each type should be used.

4.1 Pre-process Mechanisms

Mechanisms in this category involve changing the training data before feeding it into a machine learning algorithm. Preliminary mechanisms, such as the ones proposed by Kamiran and Calders (2012) and Luong, Ruggieri, and Turini (2011), proposed changing the labels of some instances or reweighing them before training to make the classification fairer. Typically, the labels that are changed are related to samples that are closer to the decision boundary since these are the ones that are most likely to be discriminated. More recent mechanisms suggest modifying feature representations, so that a subsequent classifier will be fairer (Abusitta, Aïmeur, & Wahab, 2019; Calmon, Wei, Vinzamuri, Ramamurthy, & Varshney, 2017; Feldman et al., 2015; Louizos, Swersky, Li, Welling, & Zemel, 2016; Madras, Creager, Pitassi, & Zemel, 2018; Samadi, Tantipongpipat, Morgenstern, Singh, & Vempala, 2018; Xu, Yuan, Zhang, & Wu, 2018; Zemel, Wu, Swersky, Pitassi, & Dwork, 2013).

For example, Feldman et al. (2015) suggest modifying the features in the dataset so that the distributions for both privileged and unprivileged groups become similar, and therefore, making it more difficult for the algorithm to differentiate between the two groups. A tuning parameter λ was provided for controlling the trade-off between fairness and accuracy ($\lambda = 0$ indicates no fairness considerations, while $\lambda = 1$ maximizes fairness). Chierichetti, Kumar, Lattanzi, and Vassilvitskii (2017); Backurs et al. (2019) use the same notion of fair representation learning and apply it for fair clustering, and Samadi et al. (2018) apply it for fair dimensionality reduction (PCA).

Note that this approach to achieving fairness is somewhat related to the field of *data compression* (Tishby, Pereira, & Bialek, 1999; Zemel et al., 2013). It is also very closely related to *privacy* research since both fairness and privacy can be enhanced by obfuscating the sensitive information, with the adversary goal of

minimal data distortion (Edizel, Bonchi, Hajian, Panisson, & Tassa, 2019; Kazemi, Zadimoghaddam, & Karbasi, 2018).

4.2 In-process Mechanisms

These mechanisms involve modifying the machine learning algorithms to account for fairness during the training time (Kamishima, Akaho, Asoh, & Sakuma, 2012; Woodworth, Gunasekar, Ohannessian, & Srebro, 2017; Bechavod & Ligett, 2017a,b; Zafar, Valera, Gomez Rodriguez, & Gummadi, 2017b,a; Goh, Cotter, Gupta, & Friedlander, 2016; Calders & Verwer, 2010; Agarwal, Beygelzimer, Dudik, Langford, & Wallach, 2018).

For example, Kamishima et al. (2012) suggest adding a regularization term to the objective function that penalizes the mutual information between the sensitive feature and the classifier predictions. A tuning parameter η was provided to modulate the trade-off between fairness and accuracy.

Zafar et al. (2017a), Zafar et al. (2017b), and Woodworth et al. (2017) suggest adding constraints to the classification model that require satisfying a proxy for *equalized odds* (Zafar et al., 2017a; Woodworth et al., 2017) or *disparate impact* (Zafar et al., 2017b). Woodworth et al. (2017) also show that there exist difficult computational challenges in learning a fair classifier based on *equalized odds*.

Bechavod and Ligett (2017a) and Bechavod and Ligett (2017b) suggest incorporating penalty terms into the objective function that enforce matching proxies of FPR and FNR. Kamiran, Calders, and Pechenizkiy (2010) suggest adjusting a decision tree split criterion to maximize information gain between the split attribute and the class label while minimizing information gain with respect to the sensitive attribute. Zemel, Wu, Swersky, Pitassi, & Dwork (2013) combine fair representation learning with an in-process model by applying a multi-objective loss function based on logistic regression, and Louizos, Swersky, Li, Welling, & Zemel (2016) apply this notion using a variational autoencoder.

Quadrianto and Sharmanska (2017) suggest using the notion of *privileged learning*[2] for improving performance in cases where the sensitive information is available at training time but not at testing time. They add constraints and regularization components to the privileged learning support vector machine (SVM) model proposed by Vapnik and Izmailov (2015). They combine the sensitive attributes as privileged information that is known only at training time, and they additionally use a maximum mean discrepancy (MMD) criterion (Gretton, Borgwardt, Rasch, Schölkopf, & Smola, 2012) to encourage the distributions to be similar across privileged and unprivileged groups.

[2] Privileged learning is designed to improve performance by using additional information, denoted as the "privileged information" that is present only in the training stage and not in the testing stage (Vapnik & Izmailov, 2015).

Berk et al. (2017) propose a convex in-process fairness mechanism for regression tasks and use three regularization terms that include variations of individual fairness, group fairness, and a combined hybrid fairness penalty term. Agarwal, Dudik, and Wu (2019) propose an in-process minimax optimization formulation for enhancing fairness in regression tasks based on the suggested design of Agarwal et al. (2018) for classification tasks. They use two fairness metrics adjusted for regression tasks. One is an adjusted *demographic parity* measure, which requires the predictor to be independent of the sensitive attribute as measured by the cumulative distribution function (CDF) of the protected group compared to the CDF of the general population (Agarwal et al., 2019) using the *Kolmogorov–Smirnov statistic* (Lehmann & Romano, 2006). The second measure is the bounded group loss (BGL), which requires that the prediction errors of all groups remain below a predefined level (Agarwal et al., 2019).

4.3 Post-process Mechanisms

These mechanisms perform post-processing of the output scores of the classifier to make decisions fairer (Corbett-Davies et al., 2017; Dwork, Immorlica, Kalai, & Leiserson, 2018; Hardt et al., 2016; Menon & Williamson, 2018). For example, Hardt et al. (2016) propose a technique for flipping some decisions of a classifier to enhance equalized odds or equalized opportunity. Corbett-Davies et al. (2017) and Menon and Williamson (2018) similarly suggest selecting separate thresholds for each group separately, in a manner that maximizes accuracy and minimizes demographic parity. Dwork et al. (2018) propose a decoupling technique to learn a different classifier for each group. They additionally combine a *transfer learning* technique with their procedure to learn from out-of-group samples (to read more about transfer learning, see Pan & Yang, 2009). A possible approach for enhancing fairness through a post-process mechanism is utilizing a mathematical programming methodology that takes as input the results of a classifier and incorporates fairness constraints in order to make the results fairer.

4.4 Which Mechanism to Use?

The different mechanism types present respective advantages and disadvantages. Pre-process mechanisms can be advantageous since they can be used with any classification algorithm. However, they may harm the explainability of the results. Moreover, since they are not tailored for a specific classification algorithm, there is high uncertainty with regard to the level of accuracy obtained at the end of the process.

Similar to pre-process mechanisms, post-process mechanisms may be used with any classification algorithm. However, due to the relatively late stage in the learning

process in which they are applied, post-process mechanisms typically obtain inferior results (Woodworth et al., 2017). In a post-process mechanism, it may be easier to fully remove bias types such as *disparate impact*; however, this is not always the desired measure, and it could be considered discriminatory since it deliberately damages accuracy for some individuals in order to compensate others (this is also related to the controversies in the legal and economical field of *affirmative action*, see Fullinwider, 2018). Specifically, post-process mechanisms may treat differently two individuals who are similar across all features except for the group to which they belong. This approach requires the decision-maker at the end of the loop to possess the information of the group to which individuals belong (this information may be unavailable due to legal or privacy reasons).

In-process mechanisms are beneficial since they can explicitly impose the required trade-off between accuracy and fairness in the objective function (Woodworth et al., 2017). However, such mechanisms are tightly coupled with the machine learning algorithm itself.

Hence, we see that the selection of the method depends on the availability of the ground truth, the availability of the sensitive attributes at test time, and on the desired definition of fairness, which can also vary from one application to another.

Several preliminary attempts were made in order to understand which methods are best for use. The Hamilton (2017) study was a first effort in comparing several fairness mechanisms previously proposed in the literature (Kamishima et al., 2012; Feldman et al., 2015; Calders & Verwer, 2010; Zafar et al., 2017b). The analysis focuses on binary classification with binary sensitive attributes. The authors demonstrated that the performances of the methods vary across datasets, and there was no conclusively dominating method.

Another study by Roth (2018) showed as a preliminary benchmark that in several cases, in-process mechanisms perform better than pre-process mechanisms, and for other cases, they do not, leading to the conclusion that there is a need for much more extensive experiments.

A recent empirical study (Friedler et al., 2019) provided a benchmark analysis of several fairness-aware methods and compared the fairness–accuracy trade-offs obtained by these methods. The authors tested the performance of these methods across different measures of fairness and across different datasets. They concluded that there was no single method that outperformed the others in all cases and that the results depend on the fairness measure, on the dataset, and on changes in the train–test splits.

More research is required for developing robust fairness mechanisms and metrics or, alternatively, for finding the adequate mechanism and metric for each scenario. For instance, the conclusions reached when considering missing data might be very different than those reached when all information is available (Kallus, Mao, & Zhou, 2019; Martínez-Plumed et al., 2019). Kallus et al. (2019) explore the limitations of measuring fairness when the membership in a protected group is not available in the data. Martínez-Plumed et al. (2019) tested imputation strategies to deal with the fairness of partially missing examples in the dataset. They showed that rows containing missing values may be more fair than the rest and therefore suggest

imputation rather than deletion of these data. Pessach and Shmueli (2021) find that when there is an evident selection bias in the data, meaning that there is an extreme under-representation of unprivileged groups, pre-process mechanisms can outperform in-process mechanisms.

Tip: *Software packages for applying fairness mechanisms and metrics have become available in recent years. Several examples include Bantilan, 2018; Bellamy et al., 2018; Friedler et al., 2019; Saleiro et al., 2018; Sokol, Santos-Rodriguez, & Flach, 2019.*

5 Fairness-Related Datasets

In this section, we review the most commonly used datasets in the literature of algorithmic fairness. Some of these datasets are publicly available, and we further indicate this for each of these datasets in their description.

ProPublica Risk Assessment Dataset
The ProPublica dataset includes data from the COMPAS risk assessment system (see *Practitioners Guide to COMPAS*, 2012; Angwin, 2016; Larson, Mattu, Kirchner, & Angwin, 2016).

This dataset was previously extensively used for fairness analysis in the field of criminal justice risk (Berk et al., 2018). The dataset includes 6,167 individuals, and the features in the dataset include the number of previous felonies, charge degree, age, race, and gender. The target variable indicates whether an inmate recidivated (was arrested again) within two years after release from prison.

As for the sensitive variable, this dataset was previously used with two variations—the first when race was considered as the sensitive attribute and the second when gender was considered as the sensitive attribute (Bechavod & Ligett, 2017b; Calmon et al., 2017; Emelianov, Arvanitakis, Gast, Gummadi, & Loiseau, 2019; Friedler et al., 2019; Martínez-Plumed et al., 2019).

Adult Income Dataset
The Adult dataset is a publicly available dataset in the UCI repository (Dua & Graff, 2017) based on the 1994 US census data. The goal of this dataset is to successfully predict whether an individual earns more or less than 50,000$ per year based on features such as occupation, marital status, and education. The sensitive attributes in this dataset include age (Louizos et al., 2016), gender (Zemel et al., 2013), and race (Zafar et al., 2017b; Friedler et al., 2019; Martínez-Plumed et al., 2019).

This dataset is used with several different pre-processing procedures. For example, the dataset of Zafar et al. (2017b) includes 45,222 individuals after pre-processing (48,842 before pre-processing).

German Credit Dataset
The German dataset is a publicly available dataset in the UCI repository (Dua & Graff, 2017) that includes information of individuals from a German bank in 1994.

The goal of this dataset is to predict whether an individual should receive a good or bad credit risk score based on features such as employment, housing, savings, and age. The sensitive attributes in this dataset include gender (Louizos et al., 2016; Friedler et al., 2019) and age (Zemel et al., 2013; Kamiran & Calders, 2009). This dataset is significantly smaller, with only 1000 individuals with 20 attributes.

Ricci Promotion Dataset
The Ricci dataset includes the results of an exam administered to 118 individuals to determine which of them would receive a promotion. The dataset originated from a case that was brought to the United States Supreme Court (Miao, 2011; Rutherglen, 2009). The goal of this dataset is to successfully predict whether an individual receives a promotion based on features that were tested in the exam, as well as the current position of each individual. The sensitive attribute in this dataset is race.

Mexican Poverty Dataset
The Mexican poverty dataset includes poverty estimation for determining whether to match households with social programs. The data originated from a survey of 70,305 households in 2016 (Ibarrarán et al., 2017). The target feature is poverty level, and there are 183 features. This dataset was studied, for example, in Noriega-Campero, Bakker, Garcia-Bulle, and Pentland (2019). The authors studied two sensitive features: young and old families; urban and rural areas.

Diabetes Dataset
The Diabetes dataset includes hospital data for the task of predicting whether a patient will be readmitted. It is publicly available in the UCI repository (Dua & Graff, 2017). The data contain approximately 100,000 instances and 235 attributes. This dataset was studied, for example, in Edwards and Storkey (2015), where it was studied with race as the sensitive feature.

Heritage Health Dataset
The Heritage health dataset originated from a competition conducted by the United States as a competition to improve healthcare through early prediction. It includes data of 147,473 patients with 139 features. The goal of this dataset is to predict whether an individual will spend any days in the hospital during the next year (Brierley, Vogel, & Axelrod, 2011). This dataset was studied, for example, in Zemel et al. (2013), Louizos et al. (2016), and Tramer et al. (2017), where age was the sensitive feature.

The College Admissions Dataset
The College Admissions dataset was collected by the UCLA law school (Sander, 2004). It includes data of over 20,000 records of law school students who took the

bar exam. The goal of this dataset is to predict whether a student will pass the exam based on factors such as LSAT score, undergraduate GPA, and family income.

This dataset was used, for example, by Berk et al. (2017), where gender was studied as the sensitive feature, and Bechavod and Ligett (2017b), where race was studied as the sensitive feature.

The Bank Marketing Dataset
The Bank Marketing dataset is a publicly available dataset in the UCI repository (Dua & Graff, 2017; Moro, Cortez, & Rita, 2014), and it includes 41,188 individuals with 20 attributes. The task is to predict whether the client has subscribed to a term deposit service based on features such as marital status and age. It was previously investigated by Zafar et al. (2017b), where age was studied as the sensitive attribute.

The Loans Default Dataset
The Loans Default dataset includes 30,000 instances and 24 attributes of credit card users. It is publicly available in the UCI repository (Dua & Graff, 2017; Yeh & Lien, 2009). The goal is to predict whether a customer will default on payments. The features include age, gender, marital status, past payments, credit limit, and education.

This dataset was used, for example, by Bechavod and Ligett (2017b) and Yeh and Lien (2009), where gender was studied as the sensitive feature.

The Dutch Census Dataset
The Dutch Census dataset includes 189,725 instances and 13 attributes of individuals. It is publicly available in the IPUMS repository (Center, 2015). Kamiran and Calders (2012) and Agarwal et al. (2018) use this dataset with only the 60,420 individuals who are not underaged. Their goal is to predict whether an individual holds a highly prestigious occupation by using features such as gender, age, household details, location, citizenship, birth country, education, economic status, and marital status. The sensitive feature utilized is gender.

The Communities and Crimes Dataset
The Communities and Crimes dataset includes 1994 instances and 128 attributes of communities in the United States. It is publicly available in the UCI repository (Redmond & Baveja, 2002; Dua & Graff, 2017). The goal is to predict the number of violent crimes per 100,000 individuals based on features such as percentage of population by age, by marital status, by the number of children, by race, and more. Kamiran and Calders (2012) add a new sensitive attribute that represents whether the percentage of the African–American population in the community is greater than 0.06.

6 Discussion and Conclusion

In this chapter, we presented a comprehensive and up-to-date overview of the algorithmic fairness research field. We started by describing the main causes

of unfairness, followed by common definitions and measures of fairness, and the inevitable trade-offs between them. We then presented fairness-enhancing mechanisms, focusing on their pros and cons, aiming at better understanding which mechanisms should be used in different scenarios. Lastly, we reviewed commonly used fairness-related datasets.

In recent years, multiple research sub-fields of algorithmic fairness have emerged (Pessach & Shmueli, 2022). Notable sub-fields include: (i) fairness in sequential and online learning scenarios, where the data are collected over time; (ii) fairness in text analysis and natural learning process (NLP) modeling, where one of the major concerns is that the word embeddings may exhibit social biases and gender stereotypes; (iii) learning fair representations using generative adversarial networks (GANs); (iv) fairness in generating image textual descriptions; (v) fair recommender systems; and (vi) fair causal learning.

In addition to the already studied problems and the emerging ones, we identify several open challenges that should be further investigated in future research. One major challenge stems from biases inherent in the dataset. Such biases may arise, for example, when the labeling process was performed in an already unfair manner, or if there are under-represented populations in the dataset, or in the case of systematic lack of data and in particular labels. Representative datasets are very difficult to achieve, and therefore, it is crucial to devise methods to overcome these issues.

Another challenge is the proliferation of definitions and measures, fairness-related datasets, and fairness-enhancing mechanisms. It is not clear how newly proposed mechanisms should be evaluated, and in particular which measures should be considered? which datasets should be used? and which mechanisms should be used for comparison? A closely related challenge is the difficulty in determining the balance between fairness and accuracy. That is, what are the costs that should be assigned to each of these measures for evaluation purposes. Future efforts should be invested in generating a benchmarking framework that will allow a more unified and standard evaluation process for fairness mechanisms.

The interpretability and transparency of how fairness is addressed by ML algorithms is an important challenge. Such transparency is crucial to increase the understanding and trust of users in these algorithms and, in many domains, is even required by law. This need is further supported by several recent studies that have addressed the question of how devised mathematical notions of fairness are perceived by users (Srivastava, Heidari, & Krause, 2019; Grgic-Hlaca, Redmiles, Gummadi, & Weller, 2018). It turns out that users tend to prefer the simpler notion of demographic parity, probably due to the difficulty of grasping more complex definitions of fairness.

To conclude, since the use of algorithms is expanding to all aspects of our lives, demanding that automated decisions be more ethical and fair is inevitable. We should aspire to not only develop fairer algorithms, but also to design procedures to reduce biases in the data. Such procedures may rely, for example, on integrating both humans and algorithms in the decision pipeline. However, thus far, it seems that biased algorithms are easier to fix than biased humans or procedures (Mullainathan, 2019).

Acknowledgements This study was partially supported by the Koret Foundation grant for Smart Cities and Digital Living 2030.

References

Abusitta, A., Aïmeur, E., & Wahab, O. A. (2019). Generative adversarial networks for mitigating biases in machine learning systems. *arXiv preprint arXiv:1905.09972*.

Agarwal, A., Beygelzimer, A., Dudik, M., Langford, J., & Wallach, H. (2018). A reductions approach to fair classification. In *International Conference on Machine Learning* (pp. 60–69).

Agarwal, A., Dudik, M., & Wu, Z. S. (2019). Fair regression: Quantitative definitions and reduction-based algorithms. In *International Conference on Machine Learning* (pp. 120–129).

Angwin, J. (2016, May). *Machine bias — ProPublica.* Retrieved 2020-11-08, from https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

Backurs, A., Indyk, P., Onak, K., Schieber, B., Vakilian, A., & Wagner, T. (2019). Scalable fair clustering. In *International Conference on Machine Learning* (pp. 405–413).

Bantilan, N. (2018). Themis-ml: A fairness-aware machine learning interface for end-to-end discrimination discovery and mitigation. *Journal of Technology in Human Services*, *36*, 15–30.

Barocas, S., & Selbst, A. D. (2016). Big data's disparate impact. *Calif. L. Rev.*, *104*, 671.

Bechavod, Y., & Ligett, K. (2017a). Learning fair classifiers: A regularization-inspired approach. *arXiv preprint arXiv:1707.00044*, 1733–1782.

Bechavod, Y., & Ligett, K. (2017b). Penalizing unfairness in binary classification. *arXiv preprint arXiv:1707.00044*.

Bellamy, R. K., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., ... others (2018). AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*.

Berk, R., Heidari, H., Jabbari, S., Joseph, M., Kearns, M., Morgenstern, J., ... Roth, A. (2017). A convex framework for fair regression. *arXiv preprint arXiv:1706.02409*.

Berk, R., Heidari, H., Jabbari, S., Kearns, M., & Roth, A. (2018). Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, 0049124118782533.

Brierley, P., Vogel, D., & Axelrod, R. (2011). *Heritage provider network health prize round 1 milestone prize: How we did it–team 'market makers'*. ed.

Calders, T., & Verwer, S. (2010). Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, *21*, 277–292.

Calmon, F., Wei, D., Vinzamuri, B., Ramamurthy, K. N., & Varshney, K. R. (2017). Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems* (pp. 3992–4001).

Center, M. P. (2015). *Integrated public use microdata series, international: Version 6.4 [The Dutch Virtual Census of 2001].* Minneapolis: University of Minnesota. Retrieved 2019-11-10, from https://microdata.worldbank.org/index.php/catalog/2102/study-description doi: http://doi.org/10.18128/D020.V6.4

Chierichetti, F., Kumar, R., Lattanzi, S., & Vassilvitskii, S. (2017). Fair clustering through fairlets. In *Advances in Neural Information Processing Systems* (pp. 5029–5037).

Chouldechova, A. (2017). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, *5*, 153–163.

Chouldechova, A., & Roth, A. (2018). The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810*.

Corbett-Davies, S., & Goel, S. (2018). The measure and mismeasure of fairness: A critical review of fair machine learning. *arXiv preprint arXiv:1808.00023*.

Corbett-Davies, S., Pierson, E., Feller, A., Goel, S., & Huq, A. (2017). Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 797–806).

Dastin, J. (2018, October). *Amazon scraps secret AI recruiting tool that showed bias against women.* Reuters. Retrieved 2020-11-08, from https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G

Datta, A., Tschantz, M. C., & Datta, A. (2015). Automated experiments on ad privacy settings. *Proceedings on Privacy Enhancing Technologies*, *2015*, 92–112.

Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from http://archive.ics.uci.edu/ml

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., & Zemel, R. (2012). Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (pp. 214–226).

Dwork, C., Immorlica, N., Kalai, A. T., & Leiserson, M. (2018). Decoupled classifiers for group-fair and efficient machine learning. In *Conference on Fairness, Accountability and Transparency* (pp. 119–133).

Edizel, B., Bonchi, F., Hajian, S., Panisson, A., & Tassa, T. (2019). FaiRecSys: mitigating algorithmic bias in recommender systems. *International Journal of Data Science and Analytics*, 1–17.

Edwards, H., & Storkey, A. (2015). Censoring representations with an adversary. *arXiv preprint arXiv:1511.05897*.

Emelianov, V., Arvanitakis, G., Gast, N., Gummadi, K., & Loiseau, P. (2019). The price of local fairness in multistage selection. *arXiv preprint arXiv:1906.06613*.

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., & Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 259–268).

Friedler, S. A., Scheidegger, C., & Venkatasubramanian, S. (2016). On the (im) possibility of fairness. *arXiv preprint arXiv:1609.07236*.

Friedler, S. A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E. P., & Roth, D. (2019). A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (pp. 329–338).

Fullinwider, R. (2018). Affirmative action. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Summer 2018 ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/sum2018/entries/affirmative-action/.

Goh, G., Cotter, A., Gupta, M., & Friedlander, M. P. (2016). Satisfying real-world goals with dataset constraints. In *Advances in Neural Information Processing Systems* (pp. 2415–2423).

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, *13*, 723–773.

Grgic-Hlaca, N., Redmiles, E. M., Gummadi, K. P., & Weller, A. (2018). Human perceptions of fairness in algorithmic decision making: A case study of criminal risk prediction. In *Proceedings of the 2018 World Wide Web Conference* (pp. 903–912).

Hamilton, E. (2017). *Benchmarking four approaches to fairness-aware machine learning* (Unpublished doctoral dissertation). Haverford College.

Hardt, M., Price, E., & Srebro, N. (2016). Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems* (pp. 3315–3323).

Holstein, K., Wortman Vaughan, J., Daumé III, H., Dudik, M., & Wallach, H. (2019). Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (p. 600).

Ibarrarán, P., Medellín, N., Regalia, F., Stampini, M., Parodi, S., Tejerina, L., ... others (2017). How conditional cash transfers work. *IDB Publications (Books)*.

Joseph, M., Kearns, M., Morgenstern, J. H., & Roth, A. (2016). Fairness in learning: Classic and contextual bandits. In *Advances in Neural Information Processing Systems* (pp. 325–333).

Kallus, N., Mao, X., & Zhou, A. (2019). Assessing algorithmic fairness with unobserved protected class using data combination. *arXiv preprint arXiv:1906.00285*.

Kamiran, F., & Calders, T. (2009). Classifying without discriminating. In *2009 2nd International Conference on Computer, Control and Communication* (pp. 1–6).

Kamiran, F., & Calders, T. (2012). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, *33*, 1–33.

Kamiran, F., Calders, T., & Pechenizkiy, M. (2010). Discrimination aware decision tree learning. In *2010 IEEE International Conference on Data Mining* (pp. 869–874).

Kamishima, T., Akaho, S., Asoh, H., & Sakuma, J. (2012). Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 35–50).

Kazemi, E., Zadimoghaddam, M., & Karbasi, A. (2018). Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints. In *International Conference on Machine Learning* (pp. 2549–2558).

Kleinberg, J., Mullainathan, S., & Raghavan, M. (2017). Inherent trade-offs in the fair determination of risk scores. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*.

Larson, J., Mattu, S., Kirchner, L., & Angwin, J. (2016, May). *How we analyzed the COMPAS recidivism algorithm.* Retrieved 2020-11-08, from https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm

Lehmann, E. L., & Romano, J. P. (2006). *Testing statistical hypotheses*. Springer Science & Business Media.

Lipton, Z. C., Chouldechova, A., & McAuley, J. (2017). Does mitigating ML's disparate impact require disparate treatment? *Stat*, *1050*, 19.

Louizos, C., Swersky, K., Li, Y., Welling, M., & Zemel, R. (2016). The variational fair autoencoder. *International Conference on Learning Representations (ICLR)*.

Luong, B. T., Ruggieri, S., & Turini, F. (2011). k-NN as an implementation of situation testing for discrimination discovery and prevention. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 502–510).

Madras, D., Creager, E., Pitassi, T., & Zemel, R. (2018). Learning adversarially fair and transferable representations. *arXiv preprint arXiv:1802.06309*.

Martínez-Plumed, F., Ferri, C., Nieves, D., & Hernández-Orallo, J. (2019). Fairness and missing values. *arXiv preprint arXiv:1905.12728*.

Menon, A. K., & Williamson, R. C. (2018). The cost of fairness in binary classification. In *Conference on Fairness, Accountability and Transparency* (pp. 107–118).

Miao, W. (2011). Did the results of promotion exams have a disparate impact on minorities? using statistical evidence in Ricci v. *DeStefano. J. of Stat. Ed*, *19*.

Moro, S., Cortez, P., & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, *62*, 22–31.

Mullainathan, S. (2019, Dec). *Biased algorithms are easier to fix than biased people.* New York Times. Retrieved 2020-11-08, from https://www.nytimes.com/2019/12/06/business/algorithm-bias-fix.html?smid=nytcore-ios-share

Mutlu, O., & Moscibroda, T. (2008). Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared DRAM systems. In *2008 International Symposium on Computer Architecture* (pp. 63–74).

Noriega-Campero, A., Bakker, M. A., Garcia-Bulle, B., & Pentland, A. (2019). Active fairness in algorithmic decision making. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (pp. 77–83).

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*, 1345–1359.

Pandey, S., & Shanker, U. (2018). CART: A real-time concurrency control protocol. In *Proceedings of the 22nd International Database Engineering and Applications Symposium* (pp. 119–128).

Pessach, D., & Shmueli, E. (2021). Improving fairness of artificial intelligence algorithms in Privileged-Group Selection Bias data settings. *Expert Systems with Applications*, 185, 115667. Elsevier.

Pessach, D., & Shmueli, E. (2022). A review on fairness in machine learning. *ACM Computing Surveys (CSUR)*, 55(3), 1–44. ACM New York, NY.

Pleiss, G., Raghavan, M., Wu, F., Kleinberg, J., & Weinberger, K. Q. (2017). On fairness and calibration. In *Advances in Neural Information Processing Systems* (pp. 5680–5689).

Practitioners guide to COMPAS. (2012). Northpointe Inc. Retrieved from https://njoselson.github.io/pdfs/FieldGuide2_081412.pdf

Quadrianto, N., & Sharmanska, V. (2017). Recycling privileged learning and distribution matching for fairness. In *Advances in Neural Information Processing Systems* (pp. 677–688).

Redmond, M., & Baveja, A. (2002). A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, *141*, 660–678.

Roth, D. (2018). *A comparison of fairness-aware machine learning algorithms*. (Unpublished doctoral dissertation). Haverford College.

Rutherglen, G. (1987). Disparate impact under title VII: an objective theory of discrimination. *Va. L. Rev.*, *73*, 1297.

Rutherglen, G. (2009). Ricci v DeStefano: Affirmative action and the lessons of adversity. *The Supreme Court Review*, *2009*, 83–114.

Saleiro, P., Kuester, B., Stevens, A., Anisfeld, A., Hinkson, L., London, J., & Ghani, R. (2018). Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577*.

Samadi, S., Tantipongpipat, U., Morgenstern, J. H., Singh, M., & Vempala, S. (2018). The price of fair PCA: One extra dimension. In *Advances in Neural Information Processing Systems* (pp. 10976–10987).

Sander, R. H. (2004). A systemic analysis of affirmative action in American law schools. *Stan. L. Rev.*, *57*, 367.

Simonite, T. (2015, July). *Probing the dark side of Google's ad-targeting system.* MIT Technology Review. Retrieved 2023-04-07, from https://www.technologyreview.com/2015/07/06/110198/probing-the-dark-side-of-googles-ad-targeting-system/

Sokol, K., Santos-Rodriguez, R., & Flach, P. (2019). Fat Forensics: A Python toolbox for algorithmic fairness, accountability and transparency. *arXiv preprint arXiv:1909.05167*.

Srivastava, M., Heidari, H., & Krause, A. (2019). Mathematical notions vs. human perception of fairness: A descriptive approach to fairness for machine learning. *arXiv preprint arXiv:1902.04783*.

Tishby, N., Pereira, F. C., & Bialek, W. (1999). The information bottleneck method. In *Proc. 37th Annual Allerton Conference on Communications, Control and Computing* (pp. 368–377).

Tramer, F., Atlidakis, V., Geambasu, R., Hsu, D., Hubaux, J.-P., Humbert, M., ... Lin, H. (2017). FairTest: Discovering unwarranted associations in data-driven applications. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)* (pp. 401–416).

Vapnik, V., & Izmailov, R. (2015). Learning using privileged information: similarity control and knowledge transfer. *Journal of Machine Learning Research*, *16*, 2.

Verma, S., & Rubin, J. (2018). Fairness definitions explained. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)* (pp. 1–7).

Woodworth, B., Gunasekar, S., Ohannessian, M. I., & Srebro, N. (2017). Learning non-discriminatory predictors. In *Conference on Learning Theory* (pp. 1920–1953).

Xu, D., Yuan, S., Zhang, L., & Wu, X. (2018). FairGAN: Fairness-aware generative adversarial networks. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 570–575).

Yeh, I.-C., & Lien, C.-h. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, *36*, 2473–2480.

Zafar, M. B., Valera, I., Gomez Rodriguez, M., & Gummadi, K. P. (2017a). Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web* (pp. 1171–1180).

Zafar, M. B., Valera, I., Gomez Rodriguez, M., & Gummadi, K. P. (2017b). Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics* (pp. 962–970).

Zemel, R., Wu, Y., Swersky, K., Pitassi, T., & Dwork, C. (2013). Learning fair representations. In *International Conference on Machine Learning* (pp. 325–333).

Zimmer, M. J. (1995). Emerging uniform structure of disparate treatment discrimination litigation. *Ga. L. Rev.*, *30*, 563.

Privacy-Preserving Data Mining (PPDM)

Ron S. Hirschprung

1 Introduction

Contemporary information systems aggregate a massive amount of data that are often used deductively. While these data may yield significant benefits, they also introduces a privacy violation threat. A good example that demonstrates this phenomenon is the medical information research field, when sensitive information on patients is collected, stored in a database (DB), and analyzed. This DB is a precious source for medical research, thus may save life, and however can also violate an individual privacy that may result in shame or an increase of the health insurance rate if falling into the wrong hands. The above example demonstrates the inherent trade-off between the gain and the loss when concerning privacy issues (Rastogi et al., 2007).

Privacy is perceived as a human right in many societies, and in all democracies, governments regulate privacy in various degrees of intrusiveness and with various regulatory mechanisms (Newman, 2008). Privacy protection laws are usually general, for example, the GDPR (General Data Protection Regulation) made by the EU and went into effect from 2016 (EUR-Lex, 2016). However, while the GDPR as its name suggests is a general law, it also contains some sections that address the issue of data mining in a more direct way. For example, Art. 13 of the GDPR requires that: "...the controller shall, at the time when personal data are obtained, provide the data subject with the following further information necessary to ensure fair and transparent processing:... (f) the existence of automated decision-making, including profiling, ..., at least in those cases, meaningful information about the logic involved, as well as the significance and the envisaged consequences of such

R. S. Hirschprung (✉)
Ariel University, Ariel, Israel
e-mail: ronyh@ariel.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_38

processing for the data subject." Thus, even when privacy collides with lifesaving that is by all means a supreme value, privacy has a significant weight that must be considered. In most cases, less dramatic values than lifesavings are at stake, and privacy weight becomes even more significant. Moreover, privacy is a basic human need (Altman, 1976). If an individual privacy is compromised, he may refuse to cooperate, resulting in the stall of beneficial processes. For example, if an e-commerce transaction potentially holds a threat to privacy, one might choose an alternative way of purchasing, even at a higher price (Chiu et al., 2014). In this chapter, privacy is related to individuals and not to organizations, a common approach throughout the vast majority of researches and non-academic publishers.

The more the information released is detailed, the higher the privacy risk is. The common idiom "God is in the details" should be "The devil is in the details" from a privacy-preserving point of view. A naive approach to this problem may require that identifiers (e.g., user name) will be omitted from the published data. This approach creates a fault anonymity. Considering the massive amount of data, the existence of auxiliary data sources (which are not controlled by the publisher) that can be crossed with the published data, and the **availability of data mining (DM)** algorithms as well as high computational power, re-identification and exposure of the individual becomes a real threat. Thus, DM is one of the most powerful tools that may violate privacy and the methodologies that take privacy into account when applying DM algorithms, in other words performing an action or taking measures for data privatizing are called privacy-preserving data mining (PPDM[1]). In the literature, data mining and machine learning are used interchangeably (Mohassel and Zhang, 2017), and in most cases, PPDM methodologies apply as well to machine learning. PPDM aims to enable carrying knowledge discovery from data (KDD), which is the core goal of DM, while preserving privacy. However, since those two objectives are in a trade-off relationship, PPDM techniques actually provide the ability to mitigate privacy loss while achieving the KDD goals, and also the ability to tune the accuracy of KDD vs. privacy loss. PPDM may also be defined from another point of view: The strive to publish microdata (information at the level of the individual), which will enable DM process to discover general patterns, without revealing anything on the individual. We used the term "strive" because "revealing anything" is a utopian reality. The comprehensive process (which in most cases is only partially implemented) of PPDM on a centralized source data is depicted in Fig. 1. The green area stands for the core of the DM process under some PPDM measures. The path of privatizing starts with the raw data that are sanitized.[2] Sanitation is one of the common actions in PPDM, when the data and/or the results are altered in a way that increase privacy but still enable the DM task (e.g., some attributes are omitted).

[1] Not to be confused with "Professional Petroleum Data Management" Association, which is a non-related concept.

[2] The term "sanitation" is used here for all PPDM techniques on centralized DB (i.e., exclude cryptography), for example, adding a random noise to the data. The literature is not consistent regarding this term, and sometimes, it relates only to some of the methods.

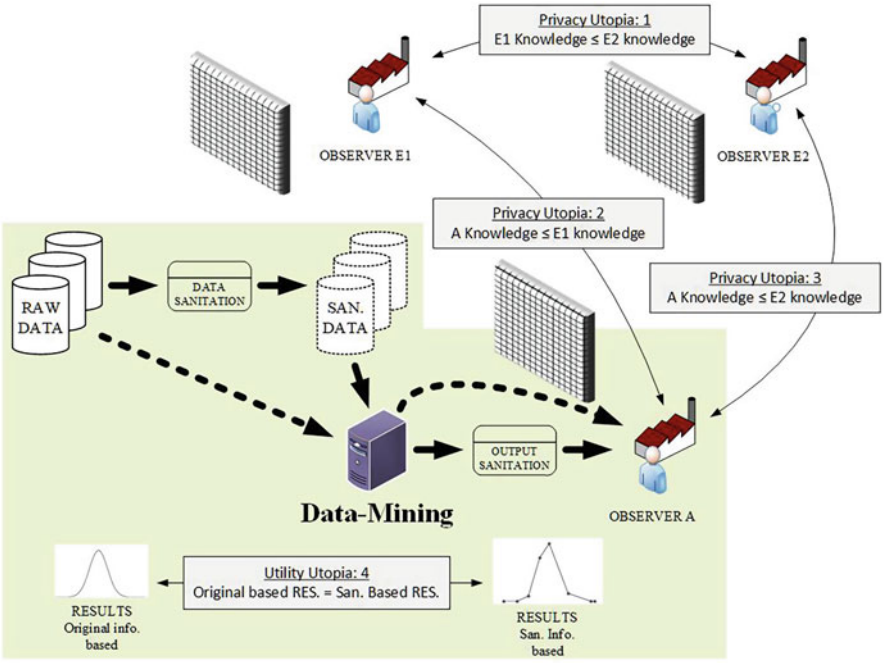


Fig. 1 The process of privacy-preserving data mining process (PPDM), demonstrated with centralized data source

Then, the sanitized data are processed through the DM mechanism, and the output can also be sanitized (e.g., hiding some association rules) before revealed to the end user (a person or an organization). One of both sanitations may be skipped as indicated by the dashed line. The masking partition depicts data hiding: Observer A is exposed to the result of the DM process only, and raw and sanitized data are hidden from him. Observer E1 (external) is exposed only to the sanitized data, and Observer E2 (external) is not exposed to the raw neither not to the sanitized data.

The figure also depicts few utopias: The first utopian situation will be if the knowledge of observer E1 about individuals is not greater than the knowledge that observer E2 has. In other words, the release (data that are published to a non-trusted entity) of the sanitized data contributes nothing to privacy discourse. The second utopia is that observer A's knowledge about individuals is not greater than the knowledge that observer E1 has. In other words, the DM process contributes nothing to privacy discourse. The third utopia is that observer A's knowledge about individuals is not greater than the knowledge of observer E2. In other words, the results of the DM process contributed nothing to the knowledge about the individual. This is the highest level of PPDM we can desire to. Finally, the forth utopia deals not with the privacy loss, but with the utility decrease. We strive to accept results from the sanitized data with the same accuracy level as from the raw data.

This last utopia suggests an inherent trade-off that may introduce significant ethical issues, e.g., when lifesaving and privacy values collide (see the example in Sect. 7). The issue of ethic in DM is controversial and lately made headlines with the Facebook–Cambridge Analytica data scandal (Chang, 2018). Cambridge Analytica purchased from Facebook private data of over 50 million US citizens without their consent, purposed to be used for political targets. In this chapter, we will not deal with the ethical aspects but only provide the techniques that can mitigate the trade-off and can also be used by developers to line up with policy-makers' requirements or regulations.

Finally, it is important to distinct between privacy and security (of the data). Security deals with the question of how do we protect our data, which is not necessarily sensitive from an individual mental point of view (e.g., credit-card details). On the other hand, privacy deals with protecting such a sensitive data (e.g., purchase history). Privacy introduces a dimension that does not exist in security: while with security we strive for total defense, with privacy we are ready to give up some of it, in order to gain some benefits. This is the inherent trade-off mentioned above. Thus, privacy and security protection methods may share the same or similar techniques but serve different goals.

2 PPDM Classification

PPDM is classified in the literature according to a few dimensions such as the knowledge that the data owner has on the expected analysis, the nature of the protection procedure, the number of data sources, the stage at which protection takes place, etc. There are no common conventions regarding PPDM classification, and various parameters or hierarchies may be applied. Anyway, while it is not really important to seek for the "true" classification, the purpose of this section is to draw a comprehensive picture of PPDM common approaches, methodologies, and techniques. Because DM is an on-the-rise prevalent tool, and privacy is at the top of list of concerns, PPDM adoption is widespread. The literature thus, as in many other domains, can be categorized into: general reviews of the subject, e.g., (Aggarwal and Philip, 2008; Agrawal and Srikant, 2000), focused on specific subdomains, e.g., (Nabar et al., 2008), and researches of new ideas and methodologies that are usually an extension of existing ones, e.g., (Holohan et al., 2017).

The general architecture of the PPDM system is classified by the location where privatizing takes place: (a) **Trusted Third-Party Analyzer:** When a trusted third party (a centralized server) exists, each data source transfers all its data to the server without privatizing. This server usually sanitizes the data before publishing a release to a non-trusted analyzer, or carries out the DM process and sanitizes the results before publishing them. The most common approaches to achieve privacy under this model are anonymization, randomization, and privatizing results; (b) **Distributed Analysis:** When there is no trusted third party, each data source privatizes its data before publishing, and the DM process can be carried out everywhere. The most

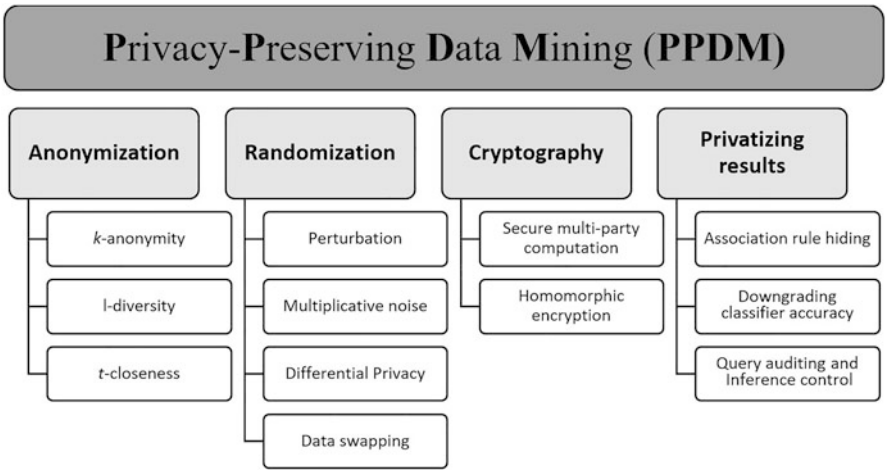


Fig. 2 Privacy-preserving data mining (PPDM) major approaches (gray boxes) and methodologies (white boxes)

common approach to achieve privacy under this model is cryptography, as parties cannot necessarily trust each other.

Considering the nature of the privatization process, i.e., the PPDM type, four major approaches are in use: (a) **Anonymization:** Parts of the data are omitted or generalized in order to avoid uniqueness. This approach aims to prevent re-identification of an individual in the dataset; (b) **Randomization:** Data are altered with some random-based operators that yield results with no noticeable laws to a bystander. This approach aims to prevent learning sensitive data about the individual, even if re-identified; (c) **Cryptography:** Data are encrypted. This approach aims to handle distributed data sources that cannot trust each other; (d) **Privatizing results:** The privatization process is applied to the results of the DM process (rather than to the raw data). The aim of this approach is to minimize the DM utility reduction resulted from the privatization process. These major approaches and the main methodologies under each one of them are depicted in Fig. 2.

2.1 Data Attributes Classification

When datasets are handled from privacy point of view (in general, and specifically in the PPDM field), it is common to classify attributes into three types:

(a) **Key attributes**: Attributes that uniquely identify the data owner, e.g., I.D. number, social security number.[3]

[3] Some attributes that are not unique by definition but close to uniqueness are also considered key attributes, e.g., the tuple first and last name.

(b) **Quasi-identifiers**: Attributes that are not unique, but can be used to re-identify the data owner, e.g., age, home town.
(c) **Sensitive attributes**: Attributes containing sensitive information, e.g., blood pressure, salary.

3 Anonymization

Uniqueness of an individual in a DB exists when one or a combination of a few quasi-identifiers are unique (have only one occurrence). Anonymization approaches aim to "hide" an individual among a group of others by reducing uniqueness, i.e., changing the dataset so that each individual's quasi-identifiers (or their combination) are identical or "similar" to those of other individuals. The first step toward anonymization will be to remove the key attributes or alternatively to generalize them (explained below) in order to lose uniqueness. However, as mentioned above, since quasi-identifiers may introduce uniqueness (e.g., an individual is the only one in the dataset with a specified birthday), more steps are required to reach anonymization. This is done by using one or both of two major techniques:

(a) **Generalization**—Replacing a relatively accurate attribute value with a less specific one, e.g., replacing the birth day 20/Oct/1965 with 1965 (which will join into one group all individuals that were born in 1965)
(b) **Suppression**—Replacing some values with an asterisk (*), i.e., value is actually omitted and not reported

Generalization and suppression are demonstrated in Sect. 3.1 bellow. The use of these techniques holds a "price" of a significant information loss, especially when the DB contains a large number of quasi-identifiers that sometimes called the "curse of dimensionality" (Agrawal and Srikant, 2000).

3.1 k-Anonymity

The concept of *k*-anonymity is based on the definition that a release of data is *k*-anonymous if each individual contained in the release cannot be distinguished from another at least *k-1* individuals also contained in the release. It was first introduced by Latanya Sweeney and Pierangela Samarati (1998). Table 1 demonstrates this concept: The left subtable (a) is the original dataset with *k*-anonymity = 1. The right subtable (b) is the release after *suppression* of the I.D. number (a key attribute), *generalization* of the birth date (quasi-identifier), and *suppression* of the gender (quasi-identifier) of some tuples(marked with asterix)—to achieve *k*-anonymity = 2.

To formalize *k*-anonymity, let R be a dataset with m quasi-identifiers $q_1, q_2 \ldots q_m$ for each record. If r is a record in R, then $r(q_x)$ is the value of attribute q_x of this record. R complies with *k*-anonymity *iff*:

Table 1 An example of k-anonymity

I.D.	Birth date	Gender	Blood pressure
457,894	20/10/65	M	140/90
946,588	2/9/65	F	120/80
129,846	1/3/68	M	130/85
945,687	5/9/68	M	120/80

a. The release before anonymization

$\Longrightarrow$

Birth date	Gender	Blood pressure
1965	*	140/90
1965	*	140/90
1968	M	110/80
1968	M	110/80

b. The release after anonymization

$$\forall r \in R : |\{s \in R : (\forall i \in 1...n : r(q_i) = s(q_i)\}| \geq k.$$

And a suppressor O complies with k-anonymity, if for a dataset R', $O(R')$ complies with k-anonymity as defined above. Applying k-anonymity with minimal data loss is an optimization type problem and was proven to be NP-hard for $k \geq 3$. To solve the computational issue, the literature offers approximation algorithms that may for example reduce complexity to $O(k \log k)$ (Meyerson and Williams, 2004).

The k-anonymity methodology is considered a basic defense and exposed to many attacks as explained in Sect. 3.2. Thus, k-anonymization is usually a requirement but cannot stand for itself. Some alternative methods are described below, and the literature also provides variations of k-anonymity. For example, with the (α-k)-anonymity in addition to the "k" requirement, it is required that the relative frequency of the sensitive value in each equivalence class (a group of records that have the same values of the quasi-identifiers in between records) is less than or equal to α ($0 < \alpha < 1$) when α is user specified (Wong et al., 2006). While k-anonymity provides some protection against re-identification, the (α-k)-anonymity methodology is an upgrade that also provides protection concerning the relationship between sensitive attributes in the DB. This problem is also NP-hard and poses a challenge to implement, especially with large DBs.

3.2 ℓ-Diversity

When data are released, k-anonymity provides some defense against identity disclosure, but not against attribute disclosure. There are 2 major attacks that can bypass k-anonymity defense:

- **Homogeneity Attack:** When all values of the sensitive data within a group of k records that have the same quasi-identifiers are the same (or have a common characteristic, like greater than a specific value), the sensitive data of a person who belong to this group can be inferred. For example, in Table 1, although the record of the person with I.D. 457894 is 2-anonymized, since all persons in his group have blood pressure above the norm (120/80), we can deduce that if a person is in this table, and born in 1965, he has high blood pressure.

- **Background Knowledge Attack:** This attack is based on a background knowledge that is usually public and not included in the released dataset. The attacker can use this auxiliary knowledge to disclose sensitive data. For example, it has been shown that based on the well-known fact that "Japanese have an extremely low incidence of heart disease," if we know that a Japanese person is included in the release, and all records within his anonymity group (that have the same quasi-identifier values) have either heart disease or viral infection, then we can deduce with high probability that this person is suffering from a viral infection (Machanavajjhala et al., 2006).

The l-diversity methodology is an extension of *k*-anonymity and aimed to address these weaknesses. When a release has classes with the same quasi-identifiers value, and each class includes at least *k* records (*k*-anonymized), l-diversity required that: "An equivalence class is said to have l-diversity if there are at least l 'well-represented' values for the sensitive attribute. A table is said to have l-diversity if every equivalence class of the table has l-diversity" (Li et al., 2007). There are few ways to address the requirement of "well-represented":

- **Distinct ℓ-diversity:** There are at least l distinct values for sensitive data in an equivalence class. However, if one value occurrence is significantly higher than another, the release is exposed to a probabilistic attack.
- **Entropy ℓ-diversity:** Entropy is a concept taken from thermodynamics, while in DM domain it represents the amount of disorder in the data. While DM seeks to decrease entropy, the higher the entropy—more privacy is gained. To define the entropy of a class *V'*, let *S* be the group of sensitive data values, and *p(V',s)* is the fraction of records in *V'* with sensitive value *s*. The entropy of *V'* is given by

$$entropy(V') = -\sum_{s \in S} p(V', s) \log p(V', s).$$

  And a table has entropy l-diversity, if for every equivalence class V', $entropy(V') > \log \mathrm{l}$.
- **Recursive (c,ℓ)-diversity:** This technique is a compromise that ensures that the most frequent value has an upper bound and the less frequent value has a lower bound on their fractions of records.

l-diversity can be sometimes difficult to achieve, especially when the distribution of the original DB is radically asymmetric. l-diversity is also sensitive to skewness attack. For example, assume an equivalence class with one sensitive data of the binary values positive/negative HIV, and a significant bias toward positive value. In this case, a person who belongs to this class will be considered to be HIV positive with high probability.

3.3 t-Closeness

Privacy loss when publishing a release can be defined by the amount of information gained by the observer as a result of receiving the release (V'). Let the observer knowledge (or belief) before receiving the release be K and the knowledge after receiving the release be K'. Now assume an intermediate hypothetical step of creating table V^m when all quasi-identifiers are removed or generalized in a way that they form a single class. The observer is now exposed only to the distribution of the sensitive data in the general population, and he cannot limit a specific person to a class. We define his knowledge as K^m, and we expect $K \leq K^m \leq K'$. While l-diversity seeks to limit the difference between K and K', t-closeness minimizes the difference between K^m and K'. The rationale behind this methodology lies on the assumption that the distribution of the sensitive data in the general population is a public knowledge anyway. The definition of t-closeness is: "An equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness" (Li et al., 2007). To measure the distance D between K^m and K', we can use one of two common metrics:

(a) The variational distance:

$$D\left[K^m, K^{'}\right] = \sum_{i=1}^{m} \frac{\left|{k^m}_i - {k^{'}}_i\right|}{2} \qquad (k \in K).$$

(b) The Kullback–Leibler (KL) distance:

$$D\left[K^m, K^{'}\right] = \sum_{i=1}^{m} {k^m}_i \log \frac{{k^m}_i}{{k^{'}}_i} = entropy\left(K^m\right) - entropy(K^{'}),$$

where $k \in K$.

Many other metrics can be found in the literature, e.g., the Earth Mover's distance (EMD) (Rubner et al., 2000). Some of them may be domain-specific, addressing a pre-known data distribution behavior.

4 Randomization

In the anonymization approach, the data in the release match the raw data, when the privatization process only reduces the resolution (generalization) or omits some of the data (suppression). On the other hand, the randomization approach allows altering the data in a way that strives to make an individual-specific record irrelevant

as a source of sensitive knowledge about this individual. Yet, even when data were altered, the goals of the DB release may still be achieved. For example, assume we would like to know the average height of people in a group. By adding some random symmetric distributed noise to the height attribute in a released dataset (perturbation), even if an individual is re-identified, the observer cannot know his height. However, if the dataset is large enough, the average calculated from the release will have no significant difference from the one calculated from the original one. A more related example to DM can be found in building a decision tree. It has shown that it is possible to build classifiers with the perturbed data (the release) with the same accuracy as of the classifiers built with the original data (Agrawal and Srikant, 2000).

4.1 Perturbation

Perturbations methodology is based on adding some random noise to the sensitive data. To formalize this technique, assume an original dataset V with n records, containing the values $(x_1\ ,\ x_2\ ,\ \ldots,\ x_n)$ for a specific attribute. A random noise vector $(y_1\ ,\ y_2\ ,\ \ldots,\ y_n)$ is drawn independently from a probability distribution F_Y. The perturbed values will be $w_i = x_i + y_i$, i.e., $(x_1 + y_1\ ,\ x_{2+}y_2\ ,\ \ldots,\ x_n + y_n)$.[4] The original data can be materialized as a set of independent random variables $(X_1\ ,\ X_2\ ,\ \ldots,\ X_n)$, all with equal distribution F_X. The random variable W of the released dataset is given by: $W = X + Y$, and naturally $X = Z - Y$.

The challenge of processing perturbed data is to reconstruct the original data distribution F_X from the perturb set. This can be approximated with a methodology based on iterative algorithm (Agrawal and Srikant, 2000). The distribution of each x_i can be estimated by

$$F'_{X_i}(a) \equiv \int_{-\infty}^{a} f_{X_i}(\mathrm{z} \mid X_i + Y_i = w_{1i})\ dz\ =\ \frac{\int_{-\infty}^{a} f_Y(w_i - z) f_X(z)\, dz}{\int_{-\infty}^{\infty} f_Y(w_i - z) f_X(z)\, dz}$$

(relying on Bayes rule for density function).
The estimation of the distribution F'_X is calculated by averaging the distributed functions for all X_i:

$$F'_X = \frac{1}{n}\sum_{i=1}^{n} F'_{X_i} = \frac{1}{n}\sum_{i=1}^{n} \frac{\int_{-\infty}^{a} f_Y(w_i - z)\, f_X(z)\ dz}{\int_{-\infty}^{\infty} f_Y(w_i - z)\, f_X(z)\ dz}.$$

And the density function f'_X si given by

[4] To measure the amount of privacy achieved by the noise addition, see the end of Sect. 4.2.

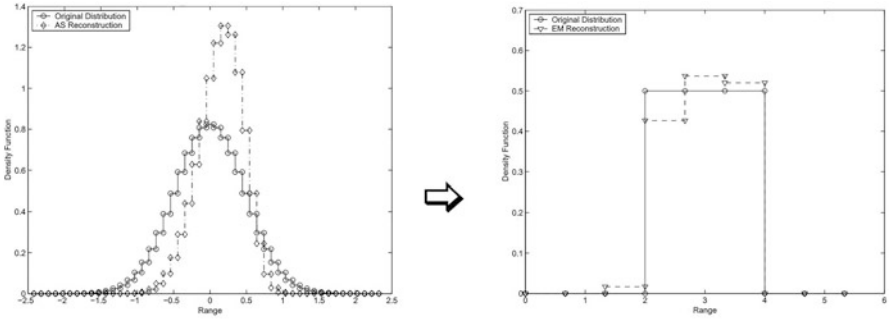
$$f'_X(a) = \frac{1}{n}\sum_{i=1}^{n}\frac{f_Y(w_i - a)f_X(a)}{\int_{-\infty}^{\infty} f_Y(w_i - z)f_X(z)\,dz}.$$

The distribution of Y can be found approximately from the released dataset by using a variety of methods such as kernel density estimation (KDE) (Wand and Jones, 1994). However, we can also approximate f_Y, and we still need f_X (or its approximation) to mind knowledge from the release. To resolve the problem, the iterative algorithm 1 may be used, with uniform distribution as an initial value:

Algorithm 1 Iterative algorithm to reconstruct the original probability of a perturbed attribute

1: $f_x^0 := Uniform$
2: $j := 0$ // Iteration number
3: **repeat**
4: $f_X^{j+1}(a) := \frac{1}{n}\sum_{i=1}^{n}\frac{f_Y(w_i-a)f_x^j(a)}{\int_{-\infty}^{\infty} f_Y(w_i-z)f_x^j(z)\,dz}$
5: $j := j + 1$
6: **until** (stopping criteria)

An example of the results of reconstructing a distribution is depicted in Fig. 3. The circle marked lines are the original distributions, while the diamond and triangle marked lines in the Gaussian (a) and uniform (b) distributions, respectively, are the reconstructed ones. It can be clearly noticed that the reconstructed distributions have the general shapes, usually the same or close center to the original distributions, but, however, deviate to some extent from the variance (Agrawal and Aggarwal, 2001).



(a. Gaussian distribution) (b. Uniform distribution)

Fig. 3 Reconstruction of the original distribution

4.2 Multiplicative Noise

The multiplicative noise methodology is a randomization approach in which rather than adding the noise (perturbation) to the original data, the value of the original data is multiplied by the noise. Multiplicative noise can be perceived intuitively as a specific technique to achieve perturbation, however usually refereed as a stand-alone methodology. Additive noise was defined above as: $w_i = x_i + y_i$, where w_i is the value in the released data , x_i the original data, and y_i the random noise. In the multiplicative methodology, $w_i = x_i \times y_i$. There are two techniques to add multiplicative noise (Kim and Winkler, 2003):

(a) **Direct multiplication**: Generate a random number y, usually from a Gaussian distribution, with a mean of μ and a small variance σ, so that: $y \sim N(\mu, \sigma)$. The noise y is usually truncated with lower and upper bounds A and B, respectively, and then re-truncated again with

$$f(y) = \frac{\frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}(y-\mu)^2\right)}{\frac{1}{\sqrt{2\pi\sigma}} \int_A^B \exp\left(-\frac{1}{2\sigma^2}(y-\mu)^2\right) de} \qquad A < y < B.$$

(b) **Logarithmic multiplication**: Calculate $t_i = \ln x_i$; let Σ be the variance–covariance matrix of T. Generate a random number e from a Gaussian distribution $N(0, c\Sigma)$ where $0 < c < 1$. Let $z_i = t_i + e_i$, and $w_i = antilog(z_i)$. The values can be negative, and in cases where this makes no sense (e.g., the number of students in a class), it is possible to add the same value to all of the data.

It is argued that multiplicative noise methodology, especially when adopting the logarithmic technique, provides higher efficiency in the DM process (more accurate results) than with noise additive (perturbation) with the same level of privacy protection (Kumar Pandya et al., 2014).

The amount of privacy achieved by supplementation of noise (with both perturbation and multiplicative methodologies) is a function of the amount of the noise. This dependency introduces a need to measure the amount of privacy achieved by the noise. One way to do so is by observing the confidence interval (CI) that was formed with a confidence level (CL), indicating the probability of the perturbed/multiplicative value being in this interval with respect to the original value. For example, assume a uniform noise of $Y \sim U[-3, 3]$ inches added to a height attribute. A CI of 5.4 in, which means that the perturbed height will fall in the range of ± 2.7 inches from the original height, has CL = 0.9. Since the deviation from the original value is with high probability, privacy is not obtained. It is important to notice that privacy is domain-dependent, e.g., 2 pounds of an adult weight is not significant, and however, 2°C difference in body temperature can be considered as a sign for illness. The method of measuring privacy based on CI/CL does not take into account the distribution of the original data, which in some cases

may result in deducting an original specific value at higher accuracy than expected (considering the CI). To resolve this problem and measure privacy in a more accurate way, a method based on the differential entropy of a random variable is offered in the literature (Aggarwal and Philip, 2008). In some cases, the observer does not receive the dataset (the release), rather he can only query it. The amount of privacy loss in this case can also be influenced by the total number of queries allowed, and not only by noise. Under this architecture, the amount of noise can be reduced to achieve more accuracy, given that the number of queries is limited. For example, it has shown that "a strong form of privacy can be maintained using a surprisingly small amount of noise—much less than the sampling error—provided the total number of queries is sub-linear in the number of DB rows" (Blum et al., 2005). This family of queries is called SuLQ (Sub-Linear Queries), and the assumption of sub-linearity is reasonable with large DBs.

4.3 Differential Privacy (DP)

The differential privacy methodology (DP) is based on measuring the effect of the presence of an individual's record in the dataset on its privacy. Assume a given dataset of m records, sanitized to provide general statistic information, but to block the access to a specific information about an individual. We can assume that if a specific person is not included in the set, his privacy is completely kept. Now, utopian situation is when adding a person to the DB, yet his privacy remains in the same level as with the set that does not include his record. Unfortunately, this is not possible, and more realistic demands will be: "The risk to one's privacy, or in general, any type of risk, such as the risk of being denied automobile insurance, should not **substantially** increase as a result of participating in a statistical database" (Dwork, 2011).

DP is quantified with the ε index ($\varepsilon \in \mathbb{R}\ ,\ \varepsilon > 0$) as follows: We define adjacent datasets as a pair of datasets that differ in only 1 record. Let $\mathcal{K}$ be a randomization function. The function $\mathcal{K}$ is said to be ε-differential if for all datasets D_1 and D_2 that are adjacent and for all the subsets of the image of $\mathcal{K}$:

$$\Pr\left[\mathcal{K}(D_1) \in S\right] \ \leq\ e^{\epsilon} \leq \Pr[\mathcal{K}(D_2) \in S].$$

And the sensitivity Δf of a function f on a dataset D is

$$\Delta f = max \| f\ (D_1) - f\ (D_2) \|_1.$$

When datasets D_1 and D_2 are all pairs of adjacent datasets in D, $\|\blacksquare\|_1$ is the l_1 norm.

A common technique used to create noise that addresses the ε-differential privacy criteria is with the Laplace distribution ($f_{Laplace}\,(x) = \exp\left(- \frac{|x|}{b} \right)$), with mean

$\mu = 0$ and standard deviation of $\sqrt{2}b$. In this case, the ε-differential of a query function f is $e^{\frac{f(D_1)-f(D_2)}{b}} \leq e^{\frac{\Delta f}{b}}$; thus, it addresses $\frac{\Delta f}{b}$-differential privacy.

The methodology of ε-differential privacy can be combined with *k*-anonymity, a framework known as (*k*-ε)-anonymity (Holohan et al., 2017). This technique decreases data loss due to the need to suppress a smaller number of records.

4.4 Data Swapping

Data randomization can be achieved not only by adding noise, but also by swapping some of the sensitive attributes across records. If for example we know that few specific records are more vulnerable than others, their sensitive attributes may be swapped with other records that are randomly selected from the table (Fienberg and McIntyre, 2004). This methodology has the advantage of preserving lower order marginal statistics (e.g., averaging attributes across a dataset). Usually, it is not a stand-alone methodology and combined with others such as *k*-anonymity. The idea was first introduced by Dalenius and Reiss (1982), who developed a theoretical framework for data swapping. While this methodology indeed preserves basic statistics, it may significantly lower the utility of extracting association rules for example, thus less relevant for PPDM.

5 Cryptography

The data in some cases may distribute among different sources that do not trust each other, so microdata cannot be simply shared. However, DM processes are based on calculating aggregated statistics and other outputs from the spread DBs. The data may be distributed in two major ways, *horizontally* and *vertically*. In horizontal distribution, each DB holds different records (or with some overlap), but with the same attributes. For example, each hospital holds information only about patients that were treated in its facility. The attributes containing the patient key attributes (e.g., full name), some quasi-identifiers (e.g., age), and sensitive attributes (e.g., HIV diagnostic) are common to all of them or may be easily transformed to create a unified single DB. In vertical distribution, each DB holds all of the records but with only part of the attributes (which may overlap). To exemplify the vertical partitioning, assume a researcher who wants to find the correlation between some disease (can be collected from hospitals) and buying trends (can be collected from retailers). Obviously, some attributes are common to both (e.g., I.D., home address), and others may be found only in one of them (e.g., blood pressure in the hospital DB and item number that was purchased in the retailer DB). Horizontal and vertical partitioning can also be mixed to create a hybrid distribution, a case in which each DB holds part of the records with part of the attributes. Cryptography is the most

common approach used to apply distributed PPDM but is not the only one. Other approaches such as randomization by perturbation can be adopted here (Liu et al., 2005); however, they have some significant limitations.

5.1 Secure Multi-party Computation

Secure multi-party computation (SMC) methodology is based on join calculations, when each participant contributes its share, but not disclosed to other participants its sensitive data (Goldreich, 1998). Among those, we can find the secure sum, for example, a problem that defines as follows: Obtain from n distributed sources the sum of inputs m_i ($i \in 1..n$), so that every source may finally know the sum $\sum_{i \in n} m_i$, but not the inputs of other sources (source i does not know m_j, where $j \in n$, $j \neq i$). This problem can be solved with the following algorithm: Assume we know an upper bound $\sum_{i \in n} m_i \leq N$. Pick one of the sources as a master, defined source number 1. The master generates a random number R distributed uniformly $R \sim U[0..N]$ and calculates $v_1 = (R + m_1) \mod N$. The value v_1 is transferred to the 2'nd source who calculates $v_2 = (v_1 + m_2) \mod N$ and back again till the last source $i = n$, who transfers his value back to the master source. Note that since R distributes uniformly in the range $[0..N]$, the value $(R + v_i) \mod N$ also distributes uniformly in this range, so source $i+1$ learns nothing about source i. Now the master source can subtract R from the value he received, to yield the required sum (Clifton et al., 2002). Other problems of this type are: (a) *Secure set union*: Each source has a set of rules, item sets, etc., and he is ready to publish its data in order to create a union with other datasets, but without revealing the owner of each set; (b) *Secure Size of Set Intersection*: Each source has a set as above, and we want to calculate the size of the intersection of these sets ($\left|\bigcap_{i \in \{1..n\}} s_i\right|$).

The implementation of SMC is based on one of two major assumptions, regarding the level of trust:

(a) **Semi-honest adversaries**: The participants obey the protocol but may try to learn more than they are supposed to. For this reason, this mode is also called "honest-but-curious."[5]
(b) **Malicious adversaries**: The participants may not obey the protocol and may even cooperate with malicious third parties.

Naturally, the level of protection is derived directly from the level of trust. To demonstrate the idea, let us take a look at *1 out of 2 oblivious-transfer* protocol, which is among of the most common protocols, and used as a basic building block for SMC (Mendes & Vilela, 2017). This protocol aims to solve the following

[5] If curiosity does not exist, the model is actually a fully honest. In this case, although the architecture is distributed—SMC becomes redundant from a privacy-preserving point of view, and it may be handled as a centralized DB.

problem: Assume Alice (the sender) has a pair of values (m_0, m_1), and Bob has a bit $\sigma \in \{0, 1\}$. The target is that by the end Bob will learn only m_σ (without knowing $m_{1-\sigma}$) and Alice will learn nothing. Under the semi-honest model, *1 out of 2 oblivious-transfer* protocol provides the following solution: The protocol is established based on the RSA asymmetric encryption. In asymmetric encryption, the receiver of the data generates one (or more) public key and a private key. The encryption can be done with the public key, and the decryption with the private key. The complexity of calculating the private key from the public key is so high that practically this task is impossible. The receiver publishes to the transmitter of the data only the public key; this way, the transmitter can encrypt the data, but an attacker, even if holds the public key (that was published), cannot decipher the encrypted message. Bob generates two RSA public encryption keys $\left(Kpub_0, Kpub_1\right)$ but knows only the private decryption key of $Kpriv_\sigma$. He sends only the public keys to Alice, and she encrypts m_0 with $Kpub_0$ and m_1 with $Kpub_1$ and sends back the encrypted values to Bob. Bob can only decrypt m_σ, and Alice does not know which of the values (m_0, m_1) was chosen. The semi-honest assumption exists here since for example, we trust Bob to know the private key of only one of the public keys he sent.

In case the semi-honest assumptions are not realistic, different protocols that address the malicious situation may be used: Alice generates public keys $Kpub = N, e$, a private key d, and a pair of random numbers (x_0, x_1) and sends the public key and the random numbers to Bob. Bob selects x_σ, generates a random number k, blinds x_σ by $v = (x_\sigma + k^e)\ mod\ N$, and sends v to Alice. Alice who does not know which of x_0 or $x_1$was chosen by Bob calculates two possible values for k: $k_0 = (v - x_0)^d\ mod\ N$ and $k_1 = (v - x_1)^d\ mod\ N$. Now Alice calculates the two values $m'_0 = m_0 + k_0$ and $m'_1 = m_1 + k_1$ and sends them to Bob. Bob calculates $m_\sigma = m'_\sigma - k$. It can be noticed that under this protocol the assumption of two partially trusting each other is not necessary.

5.2 Homomorphic Encryption

Homomorphic encryption is a methodology that enables calculations on encrypted data without the need to decrypt it first. Since the entity that carries the calculation does not have to (and cannot) decrypt the data, privacy violation may be avoided (Rivest et al., 1978). The results of the computation carried out on the encrypted data are the same as if the operations had been performed on the unencrypted data. There are two major classes of homomorphic encryption, partial and full. In partially homomorphic encryption, additive or multiplicative homomorphism enables but not both, while fully homomorphic enables both (Acar et al., 2018).

Homomorphic encryption can be used for privacy-preserving outsourced storage and computation. This allows data to be encrypted and outsourced to commercial cloud environments for processing, all while encrypted. In highly regulated industries, such as health care, homomorphic encryption can be used to enable

new services by removing privacy barriers inhibiting data sharing. For example, predictive analytics in health care can be hard to apply due to medical data privacy concerns, but if the predictive analytics service provider can operate on encrypted data instead, these privacy concerns are diminished.

6 Privatizing Data Mining Results

Even when microdata are not published, and the observer can only query the DB for statistic results, a potential of privacy violation still exists. An attacker can purposely query the DB in a way that aims to identify differentials in the dataset, and by doing so he may retrieve specific records. For example, let us have a look at Table 2:
This table includes information about peoples' names, hometown, and income. Assume we can only query for averages, but to avoid record retrieval, querying a single record is blocked. Now let us consider 2 valid queries: *Q1*) average *monthly income* of those who live in the East Coast (NY & Washington); *Q2*) average *monthly income* of everyone.

Clearly, Q1 yields: $\frac{R_1+R_2+R_3}{3} = \frac{4{,}500+6{,}500+3{,}200}{3} = 4100,$

and Q2 yields: $\frac{R_1+R_2+R_3+R_4}{3} = \frac{4500+6500+3200+6200}{3} = 4625.$

Now, let ΔQ be the average of the records included in Q2 only (not included in Q1), therefore:

$$Q_1 \cdot \frac{3}{4} + \Delta Q \cdot \frac{1}{4} = Q_2 \;\Rightarrow\; \Delta Q = \left(Q_2 - Q_1 \cdot \frac{3}{4}\right) \cdot 4 = \left(4,625 - 4,100 \cdot \frac{3}{4}\right) \cdot 4 = \mathbf{6,200},$$

and we actually retrieved Elizabeth's salary, which is a classic sensitive information!

As clearly stems, this problem is directly related to differential privacy and can be addressed with one of the above-mentioned methodologies such as randomized perturbations. However, those techniques reduce laterally the accuracy of the DM process. Assuming the DM server is trusted, i.e., only the observer is non-trusted, another approach of privatizing the results may be adopted. Rather than altering the microdata, the results of the DM process or the microdata may be altered or blocked for some queries, keeping the microdata accurate for use in other harmless queries.

Table 2 An example of retrieving microdata by querying only

Record #	Name	Hometown	Monthly income ($)
1	Dana	NY	2500
2	Andrew	Washington	6500
3	Tom	NY	3300
4	Elizabeth	San Francisco	6200

6.1 Association Rule Hiding

In DM, association rule mining is the method of discovering "interesting" relations between attributes. However, as demonstrated above, combining few association rules may disclose sensitive data. The association rule hiding methodology, first distinct 'sensitive rules' that may disclose sensitive information, and then apply methods as described herein. The concept was first introduced in 1999 as "disclosure limitation of sensitive rules" (Atallah et al., 1999), later on coined as *association rule hiding*. Association rule hiding includes two main methods: (a) **Distortion**: Altering the entry of specific records of certain queries while keeping the entry accurate for others (Oliveira et al., 2004); (b) **Blocking**: Selective removal of specific records from the DB when certain queries are applied; however, data are preserved for other queries that are believed not to compromise privacy when mining the sensitive rules (Saygin et al., 2001).

Both methods have the potential of lowering the accuracy of non-sensitive rules, thus directly affecting the level of the DM utility. The strategy of most algorithms of association rule hiding is based either on the support indication, the confidence indication, or a combination of both. Support and Confidence are used in their original definition in associating rule mining. The support of an item set X with respect to transaction t is given by $supp(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|}$, when T is the set of transactions. Intuitively, support is the proportion of transactions t in the dataset that contains the item set X. The confidence of the rule mapping datasets $X \Rightarrow Y$ is given by $conf\ (X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$. Confidence is the proportion of the transactions that contains X and that also contains Y.

6.2 Downgrading Classifiers Accuracy

Classifiers are commonly used in DM process and considered as one of its central tools. However, just like with association rules, classifiers may be used to disclose private information—even when the microdata are not disclosed. To preserve privacy when DM classification is performed, it is possible to lower the accuracy of a classifier, but keep the original data as is (Mendes and Vilela, 2017). The accuracy of a classifier is its ability to predict the class it belongs to, in other words its effectiveness. In many cases, this methodology of reducing the accuracy of the classifier can have a minor effect on the utility required but provides a significant defense against privacy violation. This is naturally an outcome of the amount of accuracy reduction and the nature of the application (its sensitivity to the accuracy). Downgrading classifier effectiveness can be viewed as a private case of association rule hiding, and thus, algorithms from this domain may be applied here.

6.3 *Query Auditing and Inference Control*

This methodology addresses directly the privacy threat described in the head of this section, when an adversary has no access to the microdata, yet with aggregated queries he may infer some knowledge at the microdata level. There are two main techniques under this methodology to provide a defense:

(a) **Query auditing**: Denying some specific queries out of all queries launched, in such a way that sensitive data will not be inferred (Chin and Ozsoyoglu, 1982). A major challenge of this methodology is that we have no prior knowledge on the sequence of the coming queries; thus the method has to be applied at real time.
(b) **Query Inference Control**: Perturbing the microdata (the data used in the DM process) or the results (output of the DM process) (Mishra and Sandler, 2006). This technique is similar to the perturbation methodology (under randomization approach); however, here the data are perturbed under the assumption that the adversary has no access to the microdata. Thus, the level of the perturbation is only required to reduce the risk of privacy violation based on the DM results.

For example, when the query operator is *sum*, for queries with k elements, when each pair of queries has at r shared elements, and the adversary already knows l values—the *query auditing* method may be applied: The number of queries must be limited to be smaller than $(2k - l + 1)/r$ in order to preserve privacy (Nabar et al., 2008).

7 Utility Assessment

Most PPDM approaches, especially randomization, reduce the data quality and thus cause some utility loss. Utility assessment is a broad issue in DM and not limited to PPDM. However, because of the inherent trade-off between utility and privacy loss, it is briefly mentioned here. There are three major indexes that evaluate the data quality: (a) *Accuracy*: A measure of how close is the secured data to the original data. (b) *Completeness*: The loss of individual data in the secured DB; and (c) *Consistency*: The loss of correlations in between records in the secured DB. A common way to measure and evaluate the utility loss is by comparing the results that the DM process yields from the original data against those from the secured data (e.g., perturbed). For example, a metric to compare the reconstructed distribution that the algorithm yields $\widehat{f_x}(x)$ with the original distribution $f_x(x)$ is based on calculating the information loss I, and given by (Agrawal and Aggarwal, 2001):

$$I\left(f_x(x)\ ,\ \widehat{f_x}(x)\right) = \frac{1}{2}E\left[\int_{\Omega_X} \left|f_x(x) -,\ \widehat{f_x}(x)\right| dx\right].$$

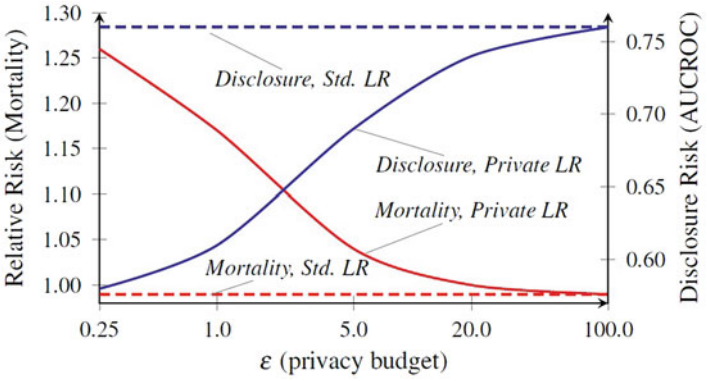


Fig. 4 The trade-off between privacy and utility in dose predicting for the Warfarin medicine

An interesting test case that demonstrates the trade-off between utility and privacy preserving is a research of pharmacogenetics (study of the role of the genome in drug response) on the medicine Warfarin (Fredrikson et al., 2014). Warfarin is an anticoagulant widely used to help preventing strokes. However, this medicine is highly dose sensitive, and an improper dose may lead to increased risk of stroke or uncontrolled bleeding. A dose can be predicted based on other patients' history and genomic information, by applying DM techniques. Naturally, genomic information of an individual is highly sensitive data, and thus, a release that includes such data is a major privacy concern. The researchers simulate analysis of the data with 5 levels of privacy (noted as *privacy budget*), and measured with ε-differential privacy. The "utility" is measured here by the mortality rate (naturally the lower the mortality—the higher the utility) and may also be referred as an accuracy index. As clearly shown in Fig. 4, increasing privacy decreases utility and vice versa.

In this example, two values collide: one of privacy preserving by protecting disclosure of sensitive health data (might be one of the most significant privacy concern) and the other of lifesaving. The inherent trade-off between privacy preserving and accuracy of the result may be reduced but cannot be completely avoided.

8 Personalized Privacy and Risk Assessment

In most of the models, privacy is treated as an important issue, but with an equal cost value to all records. However, this cost is not singular and may be varied as a function of: who owns the data, what is the nature of the data, what is the spread of distribution, etc. The value of privacy is context-dependent, for example, a specific user may consent to divulge the same personal information for different minimal rewards in different contexts (Acquisti et al., 2015). In other

words, privacy loss has some cost that under the common methodologies is fixed across all those dimensions. A relatively new methodology offers models that do not treat all records homogeneously, for example, in location-based privacy (Agir et al., 2014). Actually, most of the methodologies (if not all) mentioned above may be personalized by weighting records or a group of records according to the sensitivity to privacy. For example, the *generalization* technique mentioned above, which provides privatization (e.g., increasing *k*-anonymity), can be personalized so that different users will gain different levels of anonymity (e.g., different *k* values of *k*-anonymity) according to their preferences (Xiao and Tao, 2006). This approach prohibits the situation in which microdata are privatized according to the highest level demanded by any of the individuals included in it, and the utility of the DM process is reduced unnecessarily. To measure the level of information disclosure, we have to take into account two parameters:

(a) **The odds of inferring the sensitive data:**
Few metrics may be used to evaluate this parameter, for example, the confidence level and the entropy that were described under the randomization section.
(b) **The value of the sensitive data**:
As mentioned above, this value exists and, however, is context-dependent. The rest of this section describes a methodology to estimate this value.

Information disclosure is a loss in the process; thus the value of the sensitive data together with the odds may yield the expectancy of the cost.

The major problem in applying personalized privacy methodology is the lack of technological literacy from the user side that will enables him to understand the implication of privacy violation in a given scenario and architecture. Thus, he is unable to quantify his privacy loss. A methodology to address this issue is based on estimating the value of privacy, coined VOPE—*Value Of Privacy Estimator* (Hirschprung et al., 2016). VOPE takes an approach of simulating real-life scenario (e.g., e-commerce transaction) and an iterative converged algorithm. VOPE was tested empirically ($n = 118$) and provided bounded (converged) results on average in 74% of the cases. To validate the values received, an independent validation methodology was presented, which was proved empirically. Figure 5 depicts the values of privacy for various items in e-commerce scenario with different probabilities of disclosure. The X-axis describes various goods that were purchased with two different probabilities of disclosing this information (the fact that an individual with some quasi-identifiers performs this purchase). The value of privacy is measured with a tangible value[6] in US dollars ($), and described on the Y-axis. It can be clearly seen, for example, that the value of the information about "Adult toy" purchase is significantly higher than that for "Batteries." The whiskers in the plot indicate the standard error of the mean (SEM). It can be clearly seen that the

[6] This trait provides a significant usability in considering this cost vs. some other utilities (e.g., discount).

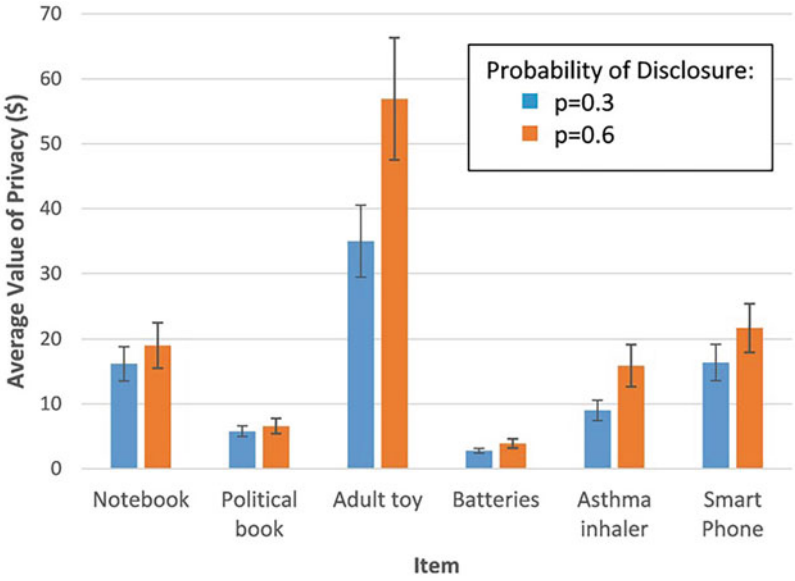


Fig. 5 Values of privacy for all items with different probabilities of disclosure

value of privacy varied across different persons, an insight that motivates the idea of personalized privacy.

9 Conclusions

This chapter starts with the description of the inherent privacy threat, directly stems from the DM process. Moreover, the utility of the DM is usually in trade-off with the amount of privacy gained. This reality introduces tough decisions of preferences between benefits, costs, and even values. The trade-off may be empowered when a conflict of interest is present, e.g., an e-commerce site collects information about individuals in order to maximize revenue, against the individual will to preserve privacy. A naive approach of just removing key attributes (e.g., full name) from the DB was found inadequate, and more sophisticated solutions are required.

To address this issue, privacy-preserving data mining (PPDM) methodologies are presented. PPDM aims to: (a) mitigate the inherent trade-off; (b) control the trade-off according to a policy. We discussed four major approaches of PPDM: **Anonymization** provides some defense against uniqueness that may lead to re-identification of the individual. The sensitive data under anonymization are not changed, and thus, in case of re-identification did occur—the individual is exposed; **Randomization** alters the sensitive data randomly, so that even if the individual record was re-identified, the sensitive data are useless at the microdata level;

Cryptography is used in case of distributed resources of data, when they cannot fully trust each other, yet a DM process should be carried on aggregated data that are ciphered; **Privatizing results** is an approach in which the DM output is privatized, while keeping the raw data intact. This approach enables later queries at the original data accuracy level by an authorized entity.

Usually, there is no single approach that provides a satisfying solution, and a combination of two or more is applied. The selection of the methodologies is derived from parameters such as: sensitivity of the data, size of the DB, distributions of quasi- and sensitive attributes in the datasets, level of trust, etc.

Finally, an important refinement of PPDM is presented, when records of different individuals (with possibly different levels of sensitivity) are not treated symmetrically, rather each individual concern and the estimation of the risk are considered.

Most researches in the field of PPDM are theoretical and are not straightforward to be implemented. The research constitutes an important novel ground for PPDM; however, further steps are required to provide privatization in the real world. Since the use of data mining and machine learning is on the rise, as well as privacy concerns and regulations, this process will most likely occur. Future algorithms will address a few opened problems and issues: (a) Better handling the trade-off between privacy protection and the utility of the DM process; (b) Dealing with high complexity of PPDM algorithms that result in consuming enormous computational power, and sometimes make the task unrealistic; and (c) Development of algorithms with higher involvement of "soft" disciplines such as psychology, to better address privacy concerns and individual's preferences.

References

Acar, A., Aksu, H., Uluagac, A. S., and Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):1–35.

Acquisti, A., Brandimarte, L., and Loewenstein, G. (2015). Privacy and human behavior in the age of information. *Science*, 347(6221):509–514.

Aggarwal, C. C. and Philip, S. Y. (2008). A general survey of privacy-preserving data mining models and algorithms. In *Privacy-preserving data mining*, pages 11–52. Springer.

Agir, B., Papaioannou, T. G., Narendula, R., Aberer, K., and Hubaux, J.-P. (2014). User-side adaptive protection of location privacy in participatory sensing. *GeoInformatica*, 18(1):165–191.

Agrawal, D. and Aggarwal, C. C. (2001). On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 247–255. ACM.

Agrawal, R. and Srikant, R. (2000). Privacy-preserving data mining. In *ACM Sigmod Record*, pages 439–450. ACM.

Altman, I. (1976). A conceptual analysis. *Environment and Behavior*, 8(1):7–29.

Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., and Verykios, V. (1999). Disclosure limitation of sensitive rules. In *Proceedings 1999 Workshop on Knowledge and Data Engineering Exchange (KDEX'99) (Cat. No. PR00453)*, pages 45–52. IEEE.

Blum, A., Dwork, C., McSherry, F., and Nissim, K. (2005). Practical privacy: the SuLQ framework. In *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 128–138. ACM.

Chang, A. (2018). The Facebook and Cambridge Analytica scandal, explained with a simple diagram. *Vox, March*, 23.

Chin, F. Y. and Ozsoyoglu, G. (1982). Auditing and inference control in statistical databases. *IEEE Transactions on Software Engineering*, Volume: SE-8(6):574–582.

Chiu, C.-M., Wang, E. T., Fang, Y.-H., and Huang, H.-Y. (2014). Understanding customers' repeat purchase intentions in B2C e-commerce: the roles of utilitarian value, hedonic value and perceived risk. *Information Systems Journal*, 24(1):85–114.

Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., and Zhu, M. Y. (2002). Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):28–34.

Dalenius, T. and Reiss, S. P. (1982). Data-swapping: A technique for disclosure control. *Journal of Statistical Planning and Inference*, 6(1):73–85.

Dwork, C. (2011). Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340.

EUR-Lex (2016). *REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL*.

Fienberg, S. E. and McIntyre, J. (2004). Data swapping: Variations on a theme by Dalenius and Reiss. In *International Workshop on Privacy in Statistical Databases*, pages 14–29. Springer.

Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D., and Ristenpart, T. (2014). Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *In 23rd USENIX Security Symposium (USENIX Security 14*, pages 17–32.

Goldreich, O. (1998). Secure multi-party computation. *Manuscript. Preliminary version*, 78.

Hirschprung, R., Toch, E., Bolton, F., and Maimon, O. (2016). A methodology for estimating the value of privacy in information disclosure systems. *Computers in Human Behavior*, 61:443–453.

Holohan, N., Antonatos, S., Braghin, S., and Mac Aonghusa, P. (2017). (k,e)-anonymity: k-anonymity with e-differential privacy. *arXiv preprint arXiv:1710.01615*.

Kim, J. and Winkler, W. (2003). Multiplicative noise for masking continuous data. *Statistics*, 1.

Kumar Pandya, B., Kumar Singh, U., Dixit, K., and Bunkar, K. (2014). Effectiveness of multiplicative data perturbation for privacy preserving data mining. *International Journal of Advanced Research in Computer Science*, 5(6).

Li, N., Li, T., and Venkatasubramanian, S. (2007). t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE.

Liu, K., Kargupta, H., and Ryan, J. (2005). Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):92–106.

Machanavajjhala, A., Gehrke, J., Kifer, D., and Venkitasubramaniam, M. (2006). l-diversity: Privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24. IEEE.

Mendes, R. and Vilela, J. P. (2017). Privacy-preserving data mining: methods, metrics, and applications. *IEEE Access*, 5:10562–10582.

Meyerson, A. and Williams, R. (2004). On the complexity of optimal k-anonymity. In *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 223–228. ACM.

Mishra, N. and Sandler, M. (2006). Privacy via pseudorandom sketches. In *Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database systems*, pages 143–152. ACM.

Mohassel, P. and Zhang, Y. (2017). SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38. IEEE.

Nabar, S. U., Kenthapadi, K., Mishra, N., and Motwani, R. (2008). A survey of query auditing techniques for data privacy. In *Privacy-Preserving Data Mining*, pages 415–431. Springer.

Newman, A. (2008). *Protectors of privacy: Regulating personal data in the global economy*. Cornell University Press.

Oliveira, S. R., Zaiane, O. R., and Saygin, Y. (2004). Secure association rule sharing. In *Advances in Knowledge Discovery and Data Mining: 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004. Proceedings* 8 (pp. 74–85). Springer Berlin Heidelberg. https://link.springer.com/chapter/10.1007/978-3-540-24775-3_10

Rastogi, V., Suciu, D., and Hong, S. (2007). The boundary between privacy and utility in data publishing. In *Proceedings of the 33rd International Conference on Very Large Databases*, pages 531–542. VLDB Endowment.

Rivest, R. L., Adleman, L., Dertouzos, M. L., et al. (1978). On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4(11):169–180.

Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The Earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121.

Samarati, P. and Sweeney, L. (1998). Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, technical report, SRI International.

Saygin, Y., Verykios, V. S., and Clifton, C. (2001). Using unknowns to prevent discovery of association rules. *ACM SIGMOD Record*, 30(4):45–54.

Wand, M. P. and Jones, M. C. (1994). Univariate kernel density estimation. In *Kernel Smoothing*, chapter 2. Chapman and Hall/CRC, Oxford.

Wong, R. C.-W., Li, J., Fu, A. W.-C., and Wang, K. (2006). (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 754–759. ACM.

Xiao, X. and Tao, Y. (2006). Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 229–240. ACM.

Explainable Machine Learning and Visual Knowledge Discovery

Boris Kovalerchuk

1 Introduction

Visual reasoning and discovery have a long history [30]. Chinese and Indians knew a visual proof of the Pythagorean Theorem in 600 B.C. Many scientists such as Bohr, Boltzmann, Einstein, Faraday, Feynman, Heisenberg, Helmholtz, Herschel, Kekule, Maxwell, Poincare, Tesla, Watson, and Watt have declared the fundamental role that *images* played in their *most creative thinking*.

1.1 Motivation

The major challenge for visual creative thinking and discovering, in multidimensional data (*n*-D data) used in ML, is that we *cannot see multidimensional data with a naked eye*. We need visual analytics tools ("*n-D glasses*") for this. The challenge starts at 4-D. Since only 2-D and 3-D data can be directly visualized in the physical 3-D world, visualization of *n*-D data becomes more difficult with higher dimensions as there is greater loss of information, occlusion, and clutter. Often, we use *non-reversible, lossy Dimension Reduction (DR)* methods such as Principal Component Analysis (PCA) that convert, say, every 10-D point into a 2-D point in visualization. While such reduction of 10 numbers to 2 numbers is valuable, in general, it is **lossy**, and producing **non-interpretable** features [49]. Thus, it can *remove key interpretable multidimensional information* before starting to learn complex *n*-D patterns. Alternative **lossless reversible methods** include General Line Coordinates

B. Kovalerchuk (✉)
Department of Computer Science, Central Washington University, Washington, DC, USA
e-mail: borisk@cwu.edu

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_39

(GLC) [22] that visualize multidimensional data losslessly without dimension reduction or loss of information during the visualization process. GLC projects the n-D points onto 2-D graphs preserving n-D information, but suffers more from occlusion [22] than lossy methods. Thus, there is great interest in the **hybrid methods**, which join the benefits of reversible and non-reversible methods for ML.

1.2 Analytical Versus Visual ML

Below we use the following terms. **Analytical ML models (AML)** are models built by *computational ML methods without using visual graphical tools.* **Visual ML models (VML)** are models built by using *visual graphical tools*. These models differ in many aspects, including generalization of ML training data, as we illustrate below with Iris data in Figs. 1 and 2. Here, Visual ML models generalize more conservatively and intuitively justified more than AML, which rather *overgeneralize.*

The **Human Visual and Verbal Learning** (**HVVL**) process from images shown in Fig. 1 leads to the following verbal description of Iris petal classes:

- Setosa – small length and small width of petal.
- Versicolor – medium length and medium width of petal.
- Virginica – large length and medium to large width of petal.

The Logistic Regression, Random Forest, Naïve Bayes, and Support Vector Machine (SVM) violate these meaningful descriptions (see Fig. 2). All of them allow long and short Versicolor Iris petals that violate the "medium" petal length and width of Versicolor (class 1 in Fig. 2). Next, all of them allow short *Virginica* Iris petal length that violates the "large" length of petal" of Virginica (class 2 in Fig. 2). The Logistic regression and Random forest allow extremely long Setosa Iris petals that violates the "short" length of Setosa (class 0 in Fig. 2).

All these ML methods dramatically overgeneralize unseen data, being quite correct on training data. This is an example of a *very typical overgeneralization* by *common ML methods*. It is a source of predictions errors for many unseen cases. Such errors are not a result of insufficient training data, but *overgeneralized task*

Fig. 1 Examples of Setosa, Versicolor, and Virginica Iris petals

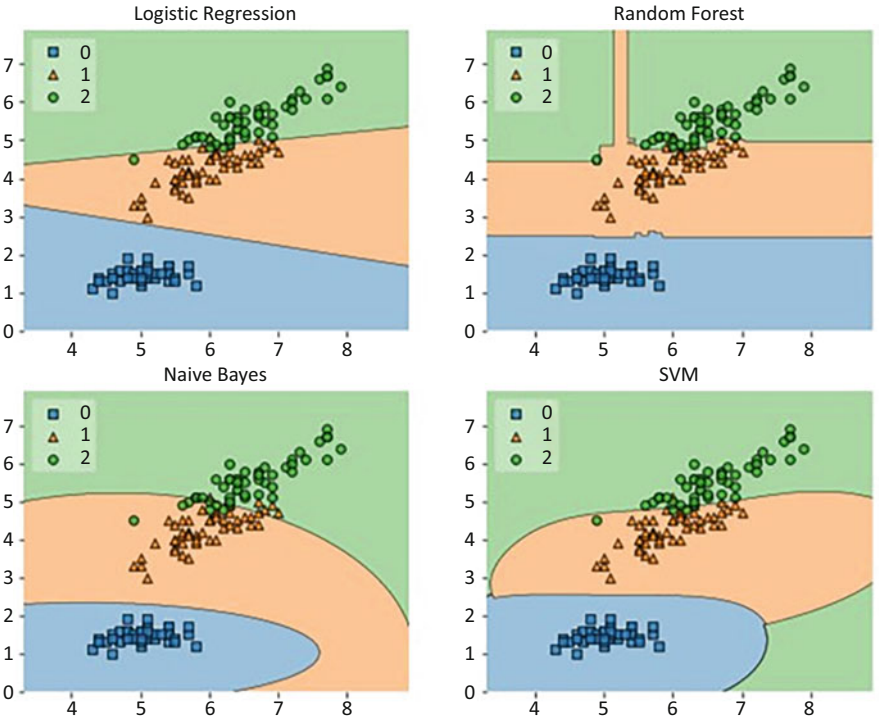


Fig. 2 Example of logistic regression classification of Iris classes. (Reproduced from ML software system [3])

formulation – discovering discrimination functions that classify *every point in the space* by Logistic Regression, Linear Discriminant Analysis, SVM, Naïve Bayes, Random Forest, Decision Trees (DT), and many other ML methods.

Figure 2 is reproduced from [3] that is a software ML package MLxtend with tools for plotting decision regions. It is surprising that not only MLxtend but many other software tools that produce such "decision regions" do not mention and do not point out how severe such "decision regions" overgeneralize classes. A more detailed analysis of these issues can be found in [23].

In contrast, the HVVL does not start with a *predefined set of models* – linear or non-linear functions, which discriminate all points in the space to three classes. It starts with *observation of the training data without any predefined model set*. It generalizes the training data more conservatively because of this observation [23].

This leads to *HVVL-inspired analytical ML models*, which use the envelopes around the training data of each class, e.g., reflecting the small length and width of petal for Iris Setosa. Such algorithms *refuse* to classify points outside of the envelopes. How can we know that we need to use an envelope? We need tools to visualize *n*-D data in 2-D without loss of *n*-D information (**lossless visualization**) allowing to see *n*-D data as we see 2-D data. In the petal example, we have a

linguistic term "*small petal*" and two corresponding 2-D visuals: a picture of the small petal in Fig. 1, and visualization of two of its attributes: length and width that form a "*small blob*" in Fig. 2. These verbal and visual representations are synergetic describing the same physical object in **intelligible** and **well-understood** ways, while both are quite uncertain and need formalization. Subjective probabilistic and fuzzy logic concepts are commonly used to formalize "small" and to "compute" and reason with words. This synergy is going further and deeper to "computing" with images [27]. The Iris petal example illustrates an important property of the **explainable ML** – it can be in *uncertain, but understandable linguistic and visual terms*, not necessarily in the formal mathematical terms.

Black Box Versus Glass Box The *Black Box models* include Deep Learning Neural Networks (DNN), Boosted Trees, Random Forests, and others. It is important to evaluate the interpretability of such methods, by comparing with human explanations, in accuracy [19]. Methods for explaining these models are surveyed in [13].

The *Glass Box models* include single Decision Trees (DT), Naive-Bayes, Bayesian Networks, Propositional and First Order Logic (FOL) rules [31], and others. Often these models are viewed as less accurate, but more intelligible, interpretable, and human understandable than black-box models. This leads to the problem of choosing between accuracy and interpretability. It is a major obstacle to the wider adoption of ML in areas with high cost of error, such as cancer diagnostics, and other domains where it is necessary to understand, validate, and trust decisions. As this chapter shows visual knowledge discovery helps getting both model accuracy and its explanation instead of choosing between them.

1.3 Approaches

Visual Analytics for Machine Learning The goals of visual analytics in ML include: aiding users in *developing* models, *understanding* how complex models work and perform, visually *debugging* models' outputs, analyzing *production*-level models, generating a *workflow* from training to production, analyzing *datasets* and model's *results*, exploring and subdividing large datasets, and comparing the *accuracy* of data groups [4, 10, 14, 18, 32, 39]. These goals belong to the stages of *developing, understanding*, *evaluating, and improving* models. The visual techniques used at these stages overlap. Heatmap is one of them. At the development stage, it visualizes *n*-D input data to get insight for selecting a class of ML models. At the model understanding stage, it highlights the salient elements identified by the model within input data, helping to understand features that the model discovered and used. At the model evaluating and improving stages, it allows seeing inconsistent salient elements to be changed.

The complementary approaches with the different *roles* of analytical, visual, black box, and glass box ML models are (1) *visualization of analytical models*, (2) *discovering analytical models* aided by *visual* methods, (3) *discovering visual mod-*

els aided by *analytical* methods, and (4) *visual explanation* of analytical models. In (1) visual methods are not involved in model creation, but help in understanding, evaluating, and improving the models. In (2) creation of the analytical ML models is guided by visual methods. In (3) analytical methods guide creation of the visual ML models. In (4) visual methods explain fully or partially the analytical models.

This chapter is organized as follows. Section 2 is devoted to visualization of trained ML models. Section 3 covers discovering *analytical* ML models aided by visual methods with case studies in Sect. 4. Section 5 is on methods to scale the process of discovering visual ML models for big data, and Sect. 6 is on methods of visual explanation of analytical models.

2 Visualization of Analytical ML Models

The goals of ML model visualizations include understanding misclassified samples, diagnosis of the model performance, model refinement, and others [34]. **Input-based** and **structure-based** are the two major approaches here. Often in the former, the model is overlaid on the data space with input data visualized. Here, **point–to-point** or **point-to-graph methods** are used to visualize data, which we discuss later.

In the structure-based approach, the model structure is visualized often along with dataflow traced in it. These visualizations are model specific, e.g., visualization of the decision tree differs from visualization of DNN [5], association rules, and the Convolutional Neural Network (CNN) for images [34]. In [34] clustering combines layers, neurons, edges, and only representative ones are visualized to decrease occlusion in visualizing CNN with thousands of neurons and millions of connections.

2.1 Matrix and Parallel Sets Visualization for Association Rules

Below we illustrate the input-based and structure-based ML model visualizations for the association rules (AR). A general AR form is $A \Rightarrow B$, where A and B are statements. Commonly A consists of several other statements, $A = P_1$ & P_2 & $\ldots$ P_k, e.g., *If customers buy both tomato (T) and cucumbers (Cu), they likely buy carrots (Ca).* Here $A = T$ & Cu and $B = Ca$.

The qualities of the AR rule are measured by *support* and *confidence* that express, respectively, *a frequency* of the itemset A in the dataset, and a portion of transactions with A and B relative to frequency of A. ARs are *interpretable* being a class of propositional rules expressed in the original domain terms.

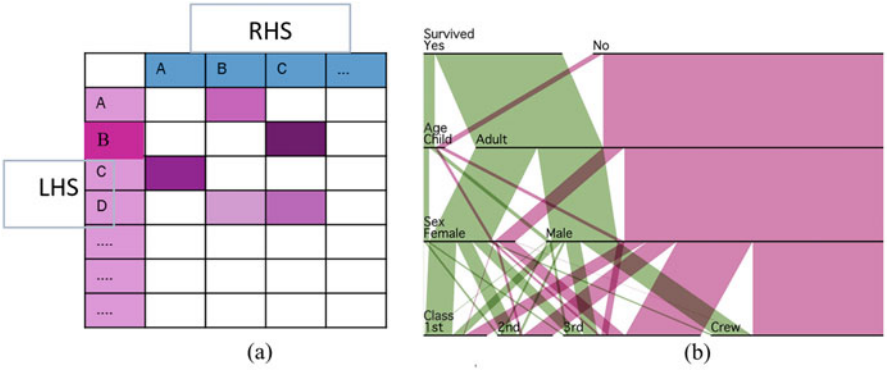


Fig. 3 Matrix and Parallel Sets visualizations of association rules: (**a**) Structure-based visualization of association rules h a matrix and heatmap. (**b**) The input-based model visualization for a set of association rules [46]

The typical questions about ARs are as follows. What are the rules with the highest support/confidence? What are outliers of the rule and their cause? Why is the rule confidence low? What is the cause of rule? What rules are non-interesting? Visualization allows answering some of these questions directly. Figure 3a shows structure-based visualization of association rules with a matrix and heatmap [46]. Here left-hand sides (LHS) of the rules are on the right and right-hand sides (RHS) of the rules are on the top. Respectively, each row shows a possible rule LHS itemset of each rule, and each column shows a possible RHS itemset. The violet cells indicate discovered rules $A \Rightarrow B$ with respective LHS and RHS. The darker color of the rule cell shows a greater rule confidence. Similarly, the darker LHS shows the larger rule support. The major challenges here are scalability and readability for a large number of LHS and RHS [46].

Figure 3b shows the input-based model visualization for a set of ARs. It uses Parallel Sets. Parallel Sets display dimensions as adjacent **parallel axes** and their values (categories) as **segments** over the axes. Connections between categories in the parallel axes form **ribbons.** The segments are similar to points and ribbons are similar to lines in Parallel Coordinates [16]. The ribbon crossings cause clutter that can be minimized by reordering coordinates and other methods [46]. Both model visualizations shown in Fig. 3 are valid for other rules-based ML models too.

2.2 Dataflow Visualization in ML Models

One of the structure-based model visualizations is model dataflow visualization, which is important for complex models such as DNN with difficulties to optimize them and understand how they work. Graph Visualizer, TensorBoard, TensorFlow's dashboard Olah's interactive essays, ConvNetJS, TensorFlow Playground, and

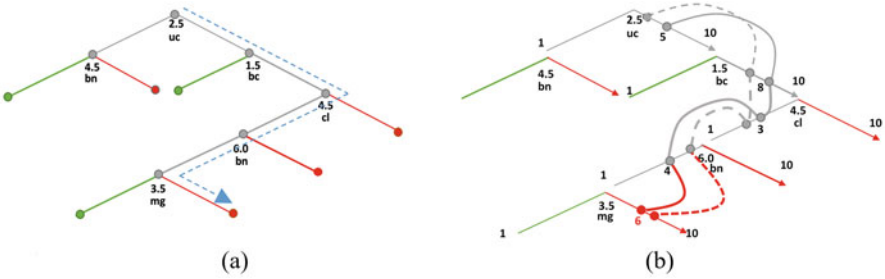


Fig. 4 DT dataflow tracing visualizations for WBC data. (**a**) Traditional visualization of WBC data decision tree. Green edges and nodes indicate the benign class and red edges and nodes indicate the malignant class. (**b**) DT with edges as Folded Coordinates in disproportional scales. The curved lines are cases that reach the DT malignant edge with different certainties due to the different distances from the threshold node

Keras are current tools for DNN dataflow visualization [45]. They allow observing scalar values, distribution of tensors, images, audio, and others to optimize and understand models by visualizing model structure at different levels of detail.

While all these tools are very useful, the major issue is that dataflow visualization itself *does not explain or optimize* the DNN model. An experienced data scientist should guide data flow visualization for this. In contrast, the dataflow for explainable models can *bring explanation* itself, as we show below for Decision Trees (DTs). Tracing the movement of a given n-D point in the DT shows all interpretable decisions made to classify this point. For instance, consider a result of tracing 4-D point $\mathbf{x} = (7, 2, 4, 1)$ in the DT through a sequence of nodes for attributes x_3, x_2, x_4, x_1 with the following thresholds: $x_3 < 5$, $x_2 > 0$, $x_4 < 5$, $x_1 > 6$ to a terminal node of class 1. The point $\mathbf{x}$ satisfies all these directly interpretable inequalities.

Figure 4a shows a traditional DT visualization for 9-D Wisconsin Breast Cancer (WBC) data from UCI Machine Learning repository [48]. It clearly presents the structure of the DT model, but without explicitly tracing individual cases. The trace is added with a dotted polyline in this figure. Figure 4b shows two 5-D points $\mathbf{a} = (2.8, 5, 2.5, 5.5, 6.5)$ and $\mathbf{b} = (5, 8, 3, 4, 6)$. Both points reach the terminal malignant edge of the DT, but with different certainty. The first point reaches it with lower certainty, having its values closer to the thresholds of uc and bn coordinates.

In this visualization, called **Folded Coordinate Decision Tree (FC-DT)** visualization, the edges of the DT not only connect decision nodes, but also serve as *Folded Coordinates* in disproportional scales for WBC data. Here, each coordinate is folded at the node threshold point with different lengths of the sides. For instance, with threshold $T = 2.5$ on the coordinate uc with the interval of values [1, 10], the left interval is [1, 2.5) and the right interval is [2.5, 10]. In Fig. 4b, these two unequal intervals are visualized with equal lengths, i.e., forming a *disproportional scale*.

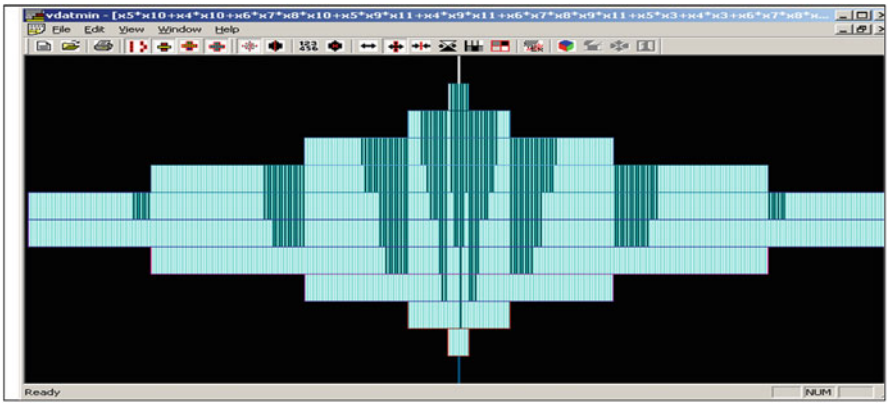


Fig. 5 Full 11-D Boolean space visualized in centered Hansel chain order with discovered classes – malignant (dark) and benign (light) breast cancer cases

2.3 Input-Based ML Model Visualization for Binary Data

This section presents the **MDF algorithm** for input-based ML model visualization for binary input n-D data [28–30]. Such data are common in medical applications with binary symptoms, where parallel line coordinates often produce heavily overlapped visualization of classes due to only two values on each binary coordinate [28, 30]. The method relies on the *monotone structural relations* between Boolean points in the n-D binary cube, E^n, and visualizes them in 2-D as chains of Boolean points located as bars in the multiple disk form (MDF). All 2048 11-D points of the E^{11} Boolean space are shown in MDF in Fig. 5, where each 11-D point is visualized as a dark bar (class 1, malignant) and light bar (class 2, benign).

To construct MDF the algorithm computes the Boolean norms of the n-D vectors – the number of 1s in each point. Then points are grouped according to these norms from 0 to n forming $n + 1$ groups. Each group occupies a strip (disk). Disks are located one on the top of another one. Then, alternative procedures assign the horizontal position to each of the n-D point on the disk.

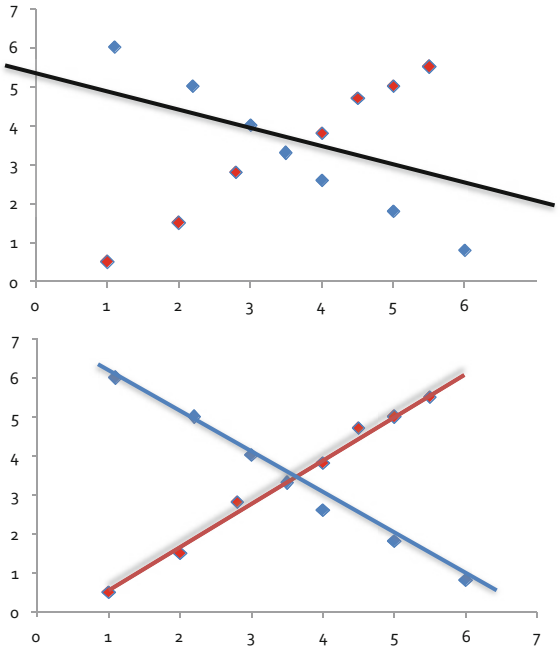
The procedure P_1 orders points according to their numerical decimal value allowing visual comparison of multiple Boolean functions. The procedure P_2 orders points according to monotone Hansel chains [29] allowing, in addition, visualizing the structure of the data along Hansel chains. The procedure P_3 orders the Hansel chains, in addition to P_2, unveiling the border between the two classes of elements, The procedure, P_4 expands Hansel chains from P_3 up and down to points absent in training data generalizing the border between classes using monotone expansion, which is first tested on the training data. Figure 5 shows the resulting directly interpretable visualization of benign and malignant classes in the original cancer features [28].

3 Discovery of Analytical ML Models Aided by Visual Methods

There is a growing trend to move from *visualization of solution* to *discovering the solution visually* [22]. The methods for discovering ML models by visual means rely on lossless and lossy methods for visualizing *n*-D data based on Parallel and Radial Coordinates, RadVis, Manifolds, t-SNE General Line Coordinates, Shifted Paired Coordinates, Collocated Paired Coordinates, and others. Methods of interactive visual model discovery leverage human *perceptual abilities* and human abilities to *adjust tasks on the fly*. Analytical ML methods enhance visual knowledge discovery by *computational means* making visual discovery easier.

As was pointed out above, the main challenge in visual knowledge discovery is that we cannot see patterns in multidimensional data by a naked eye in contrast with 2-D and 3-D spaces. Figure 6 shows an example of visual knowledge discovery in 2-D for the data in the table on the left. Here a single black line cannot discriminate these two "crossing" classes. In addition, the visualization clearly shows that any single line cannot discriminate these classes.

However, a common ML modeling practice (without visualizing the data) starts with a simplest model, which is a linear discrimination function (black line in Fig. 6) to separate the blue and red points. It will fail. In contrast, visualization immediately



x	y	class
1	0.5	1
1.1	6	2
2	1.5	1
2.2	5	2
2.8	2.8	1
3	4	2
3.5	3.3	1
4	3.8	1
4	2.6	2
4.5	4.7	1
5	1.8	2
5	5	1
5.5	5.5	1
6	0.8	2

Fig. 6 "Crossing" classes that cannot be discriminated by a single straight line

gives an insight of a correct model class of "crossing" two linear functions, with one line going over blue points, and another one going over the red points. The question is – how to reproduce such success in 2-D for *n*-D data, where we cannot see the data with a naked eye? The next section presents methods for *lossless and interpretable visualization* of *n*-D data in 2-D for this.

3.1 Approaches to Visualize Multidimensional Data

Often multidimensional data are visualized by lossy dimension reduction (e.g., Principal Component Analysis (PCA), Multidimensional Scaling (MDS), t-SNE) or by splitting *n*-D data to a set of low dimensional data (pairwise correlation plots). While splitting is useful it *destroys integrity* of *n*-D data, and leads to a shallow understanding complex *n*-D data. To mitigate splitting difficulty an additional and difficult perceptual task of assembling 2-dimensional visualized pieces to the whole *n*-D record must be solved. An alternative way for deeper understanding of *n*-D data is developing visual representations of *n*-D data in low dimensions without such data splitting and loss of information as **graphs** not 2-D points.

The examples of such visualization methods for *n*-D data are Parallel Line Coordinates (PLC) where all coordinates are parallel, Radial Line Coordinates (RLC) where all coordinate are radial and other General Line Coordinates (GLC) [22]. Figure 7 shows 6-D point $\mathbf{x} = (4, 3, 0, 5, 4, 10)$ in PLC and in RLC. The major challenges of methods like PCA, MDS, and t-SNE are in the loss of *n*-D information [35] and difficulties to interpret new coordinates, while the major challenge for GLCs is occlusion [22].

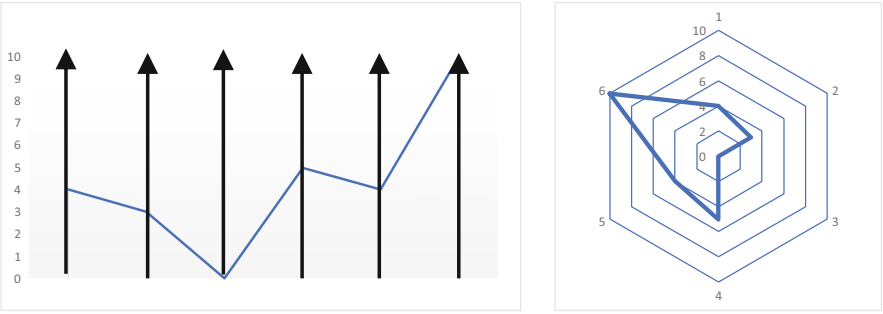


Fig. 7 6-D point $x = (4, 3, 0, 5, 4, 10)$ in PLC and 7-D point in RLC

3.2 Theoretical Limits to Preserve n-D Distances in Lower Dimensions: Johnson-Lindenstrauss Lemma

The source of the loss of information in dimension reduction from n-D to a smaller k-D is in the smaller neighborhoods in k-D. The 2-D/3-D visualization space does not have enough neighbors to represent the same n-D distances in 2-D. For instance, the 3-D binary cube has 2^3 nodes, but the 10-D hypercube has 2^{10} nodes. Mapping 2^{10} 10-D points to 2^3 3-D points leads to the distortion of n-D distances, because the variability of distances between 3-D points is much smaller than between 10-D points. It leads to the significant **corruption** of n-D distances in 2-D visualization. The Johnson-Lindenstrauss lemma states these differences explicitly. It implies that only *a small number* of *arbitrary* n-D points can be mapped to k-D points of a smaller dimension k that *preserves n-D distances with relatively small deviations.*

Johnson-Lindenstrauss Lemma [17] Given $0 < \varepsilon < 1$, a set X of m points in R^n, and a number $k > 8 \ln (m)/\varepsilon^2$, there is a linear map f: $R^n \rightarrow R^k$ such that for all $u, v \in X$.

$$(1 - \varepsilon) \parallel u - v\parallel^2 \leq \parallel f(u) - f(v)\parallel^2 \leq (1 + \varepsilon) \parallel u - v\parallel^2.$$

In other words, this lemma sets up a relation between n, k, and m when the distance can be preserved with some allowable error ε. A version of the lemma [6] defines the possible dimensions $k < n$, such that for any set of m points in R^n there is a mapping *f:* $R^n \rightarrow R^k$ with "similar" distances in R^n *and* R^k between mapped points. This similarity is expressed in terms of error $0 < \varepsilon < 1$.

For $\varepsilon = 1$ the distances in R^k are less or equal to $\sqrt{2}\,S$, where S is the distance in R^n. This means that the distance s in R^k will be in the interval [0, 1.42S]. In other words, the distances will not be more than 142% of the original distance, i.e., it will not be much exaggerated. However, it can dramatically diminish to 0. The lemma and this theorem allow to derive three formulas, to estimate the number of dimensions (sufficient and insufficient) to support the given distance errors. See Table 1. For details, see [22]. These formulas and table show that to keep distance errors within about 30%, for just 10 arbitrary high-dimensional points, the number of dimensions k needs to be over 1900 dimensions, and over 4500 dimensions for 300 arbitrary points. The point-to-point visualization methods do not meet these requirements for arbitrary datasets. Thus, the Johnson-Lindenstrauss Lemma sets up the theoretical limits to preserve n-D distances in 2-D.

3.3 Visual Knowledge Discovery Approaches

Visual knowledge discovery approaches include discovering:

Table 1 Dimensions required supporting errors within ±31%

Number of arbitrary points in *n*-D space	Sufficient dimension by formula 1	Sufficient dimension by formula 2	Insufficient dimension by formula 3
10	1974	2145	1842
20	2568	2791	2397
30	2915	3168	2721
40	3162	3436	2951
50	3353	3644	3130
60	3509	3813	3275
70	3642	3957	3399
80	3756	4081	3506
90	3857	4191	3600
100	3947	4289	3684
200	4541	4934	4239
300	4889	5312	4563

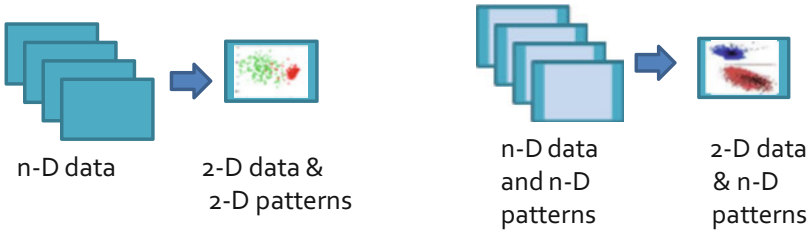


Fig. 8 Visual knowledge discovery approaches

1. **Patterns in 2-D**: *converting* *n*-D data to 2-D data and then *discovering* 2-D patterns in this visualization.
2. **Patterns in *n*-D**: *discovering* *n*-D patterns in *n*-D data then *visualizing* *n*-D patterns in 2-D as *graphs*.
3. **Patterns in 2-D and *n*-D**: some patterns are discovered in (1) with controlled errors and some are discovered in (2).

Figure 8 illustrates the first two approaches.

The patterns in 2-D approach have lossy and lossless versions. The *lossy version* includes three stages:

- **Point-to-point** lossy Dimension Reduction (DR) – converting each *n-D point to a 2-D point*.
- *Visualization* of 2-D points.
- *Interactive discovery* of 2-D patterns in *point-to-point visualization.*

The use of Principal Component Analysis (PCA) as DR with visualization of the first two principal components is an example of this approach. In general, there is no way to restore a given *n*-D point from these two principal components.

The *lossless approach* also has three stages:

- **Point to graph** lossless DR – converting each *n*-D point to a graph in 2-D (*n*-D data fully restorable from the graph).
- *Visualization* of the graph in 2-D.
- Interactive discovery of 2-D patterns on graphs in visualization.

3.4 Point-to-Point Projections

Multiple embedding techniques are in use for input-based ML model visualization, such as Principal Component Analysis, and t-Distributed Stochastic One Neighbor Embedding (t-SNE) [44]. These *point-to-point* projection methods convert *n*-D points to 2-D or 3-D points for visualization. PCA can *distort local neighborhoods* [9]. Often t-SNE attempts preserving local data neighborhoods, at the expense of *distorting global structure* [9]. Below we summarize and analyze the challenges and warnings with t-SNE highlighted in [9, 44]. One of them is that t-SNE may not help to find outliers or assign meaning to point densities in clusters. Thus, outlier and dense areas visible in t-SNE may not be them in the original *n*-D space. Despite this warning, we can see the statements that users can easily identify outliers in t-SNE [5]. It will be valid only after showing that the *n*-D metrics is not distorted in 2-D for the given data. In general, t-SNE similarity in 2-D differs from similarly in *n*-D similarly as shown in the Johnson-Lindenstrauss lemma above.

The AtSNE algorithm [11] is to resolve the difficulties of t-SNE algorithm for capturing the global *n*-D data structure by generating 2-D anchor points (2-D skeleton) from the original *n*-D data with a hierarchical optimization. This algorithm is only applicable to the cases when the global structure of the *n*-D dataset can be captured by a planar structure using point-to-point mapping (*n*-D point to 2-D point). In fact, 2-D skeleton can corrupt the *n*-D structure (see the Johnson-Lindenstrauss lemma above). Moreover, the meaningful similarity between *n*-D points can be *non-metric*. Thus, t-SNE may not *reflect n-D structures in a low-dimensional map* [44]. This is the *most fundamental deficiency of all point-to-point methods*. Therefore, we focus on **point-to-graph** GLC methods, which open a new opportunity to address this challenge.

3.5 General Line Coordinates to Convert n-D Points to Graphs

General Line Coordinates (GLC) [22, 26] break a 400-year-old tradition of using orthogonal Cartesian coordinates, which fit well to modeling the 3-D physical world, but are limited, for lossless visual representation, of the diverse and abstract high-dimensional data, which we deal with in machine learning. GLC *relaxes the requirement of orthogonality*. In GLC, the points on coordinates form **graphs**, where coordinates can overlap, collocate, be connected or disconnected, straight

Table 2 General Line Coordinates (GLC): 3-D visualization

Type	Characteristics
3-D General Line Coordinates (GLC)	Drawing n coordinate axes in 3-D in a variety of ways: curved, parallel, unparalleled, collocated, disconnected, etc.
Collocated Tripled Coordinates (CTC)	Splitting n coordinates into triples and representing each triple as 3-D point in the same three axes; and linking these points to form a directed graph. If n mod 3 is not 0 then repeat the last coordinate X_n one or two times to make it 0.
Basic Shifted Tripled Coordinates (STC)	Drawing each next triple in the shifted coordinate system by adding (1, 1, 1) to the second triple, (2, 2, 2) to the third triple $(i-1, i-1, i-1)$ to the i-th triple, and so on. More generally, shifts can be a function of some parameters.
Anchored Tripled Coordinates (ATC) in 3-D	Drawing each next triple in the shifted coordinate system, i.e., coordinates shifted to the location of the given triple of (anchor), e.g., the first triple of a given n-D point. Triple are shown relative to the anchor easing the comparison with it.
3-D Partially Collocated Coordinates (PCC)	Drawing some coordinate axes in 3-D collocated and some coordinates not collocated.
3-D In-Line Coordinates (ILC)	Drawing all coordinate axes in 3D located one after another on a single straight line.
In-Plane Coordinates (IPC)	Drawing all coordinate axes in 3D located on a single plane (2-D GLC embedded to 3-D).
Spherical and polyhedron coordinates	Drawing all coordinate axes in 3D located on a sphere or a polyhedron.
Ellipsoidal coordinates	Drawing all coordinate axes in 3Dlocated on ellipsoids.
GLC for linear functions (GLC-L)	Drawing all coordinates in 3D dynamically depending on coefficients of the linear function and value of n attributes.
Paired Crown Coordinates (PWC)	Drawing odd coordinates collocated on the closed convex hull in 3-D and even coordinates orthogonal to them as a function of the odd coordinate value.

or curvy and go to any direction. Table 2 describes 3-D GLC types. Figure 9 shows examples of several 2-D GLC types. The case studies below show the benefits of GLC for ML.

4 Case Studies on Visual Discovery

Below we show several examples of case studies with the lossless/reversible GLC approach, for ML model discovery, based on *point-to-graph methodology*.

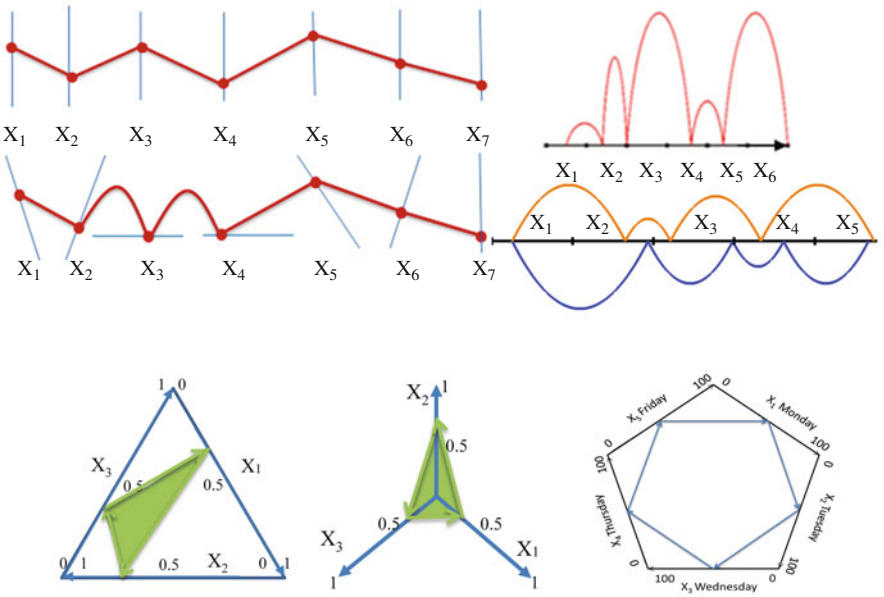


Fig. 9 Examples of different GLCs: Parallel, Non-parallel, Curved, In-line Coordinates Pentagon, Triangular, and Radial Coordinates

4.1 Case Study with GLC-L Algorithm

The first example is on the linear discrimination of 4-D data, by applying GLC-L algorithm [22, 25]. This algorithm allows both visualization of an existing trained ML model, and visual discovery of a new ML model. We start from presenting *visualization of an existing model*. GLC-L can visualize any two-class *linear or additive non-linear discrimination function*,

$$F(\mathbf{x}) = a_1 f_1(\mathbf{x}) + a_2 f_2(\mathbf{x}) + \cdots + a_n f_n(\mathbf{x}),$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Here if $F(\mathbf{x}) > T$ then $\mathbf{x}$ belongs to class 1, else $\mathbf{x}$ belongs to class 2. In a linear case, each $f_i(\mathbf{x}) = x_i$.

Figure 10 demonstrates the GLC-L for a linear model with four vectors X_1–X_4 in black. Each of them is a coordinate that form a *non-parallel unconnected coordinate system*. This figure also contains four vectors $\mathbf{x}_1$–$\mathbf{x}_4$ (in blue) located on coordinates X_1–X_4 with lengths $|\mathbf{x}_i| = x_i$. These vectors represent a 4-D point $\mathbf{x} = (x_1, x_2, x_3, x_4) = (1, 0.8, 1, 1.2)$. Thus, each blue vector $\mathbf{x}_i$ shows one of x_i from $\mathbf{x}$ and together $\mathbf{x}_1$–$\mathbf{x}_4$ represent losslessly $\mathbf{x}$.

The next step is drawing vectors $\mathbf{x}_1$–$\mathbf{x}_4$ *one after another*, to form a directed graph (see Fig. 10 on the left). Then the last point of this graph $B = A + \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + \mathbf{x}_4$ is projected onto the horizontal axis U (see a blue dotted line in Fig. 10). To simplify,

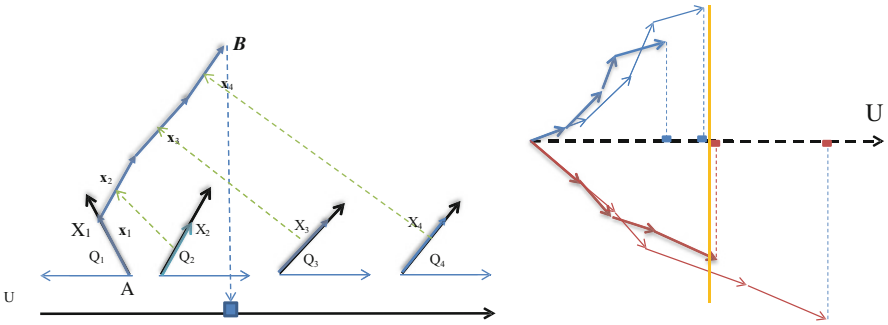


Fig. 10 GLC-L concept

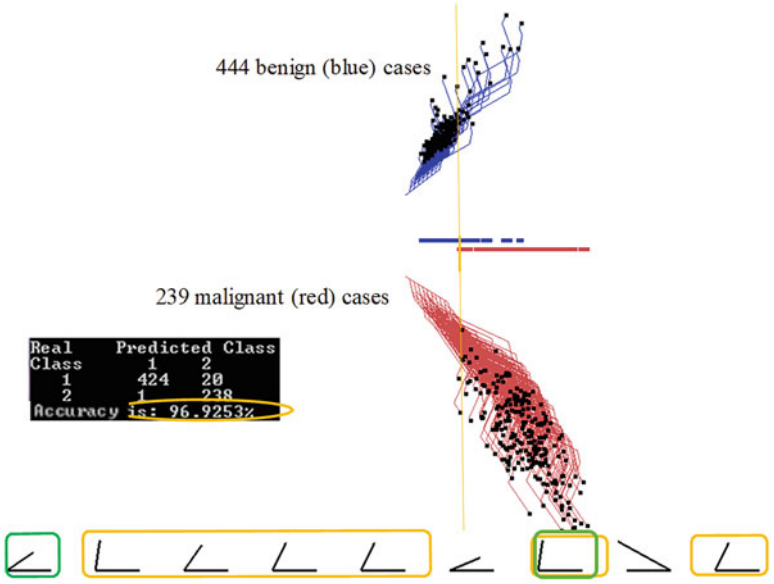


Fig. 11 GLC-L WBC data example

visualization axis U can be collocated with the horizontal lines that define the angles Q_i as shown on the left in Fig. 10. The length of the projection on coordinate U (black horizontal line) from the origin of U is a linear function $F(\mathbf{x})$.

Figure 10 on the right shows graphs of two 4-D points (in blue) and two 4-D points (in red), which are drawn mirrored relative to the axis U. The blue 4-D points belong to class 1 and red ones belong to class 2. The vertical yellow line is a threshold line, which discriminates the two classes. Figure 11 shows the application of the GLC-L algorithm for visual linear discrimination of 9-D Wisconsin Breast Cancer (WBC). The blue polylines (graphs) represent 444 benign cases and red ones 239 malignant cases with vertical yellow line discriminating these two classes.

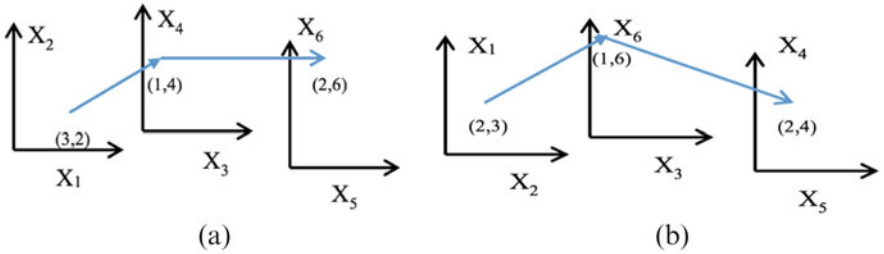


Fig. 12 6-D point **a** = (3, 2, 1, 4, 2, 6) in Shifted Paired Coordinates. (**a**) Point **a** in (X_1, X_2), (X_3, X_4), (X_5, X_6) as a sequence of pairs (3, 2), (1, 4) and (2, 6). (**b**) Point **a** in (X_2, X_1), (X_3, X_6), (X_5, X_4) as a sequence of pairs (2, 3), (1, 6) and (2, 4)

The confusion matrix shows that it misclassified 1 malignant case and 20 benign cases with the total accuracy of 96.92% [22, 25]. To get a 100% accurate classification of cancer cases, the threshold line can be moved to the left interactively. It will misclassify more benign cases, which is a less of a problem, than missing a cancer case. The angles at the bottom of Fig. 11 show the contribution of each attribute to the discrimination function $F(\mathbf{x})$. The angles with green outlines contribute the most (have larger positive coefficients in $F(\mathbf{x})$). A user can interactively adjust angles (and respective importance and contribution of attributes) to get a more meaningful and interpretable classification model. This example shows the idea of using the GLC-L algorithm beyond just visualizing the already existing ML classification model, but using GLC-L to find a better model by modifying the existing one. Alternatively, a GLC-L classification model can be built from scratch by assigning coefficients (via angles) and sliding thresholds interactively. The advantage of GLC-L is its explanatory power. It provides a *full and interpretable visual representation* of the function $F(\mathbf{x})$ without any loss of n-D information.

4.2 Case Study with the FSP Algorithm

This case study deals with the same 9-D WBC data by using the FSP algorithm [24] and Shifted Paired Coordinates (SPC) [26] for a graph representation of n-D points. The idea of SPC is presented in Fig. 12. The **Shifted Paired Coordinates** (**SPC**) visualization of the n-D data requires the splitting of n coordinates X_1–X_n into pairs producing the $n/2$ non-overlapping pairs (X_i, X_j), such as (X_1, X_2), (X_3, X_4), (X_5, X_6), . . . ,(X_{n-1}, X_n). In SPC, a pair (X_i, X_j) is represented as separate orthogonal Cartesian Coordinates (X, Y), where X_i is X and X_j is Y, respectively.

In SPC, each coordinate pair (X_i, X_j) is *shifted* relative to other pairs to avoid their overlap. This creates $n/2$ scatter plots. Next, for each n-D point $x = (x_1, x_2, \ldots, x_n)$, the point ($x_1$, x_2) in (X_1, X_2) is connected to the point (x_3, x_4) in (X_3, X_4) and so on until point (x_{n-2}, x_{n-1}) in (X_{n-2}, X_{n-1}) is connected to the point (x_{n-1},

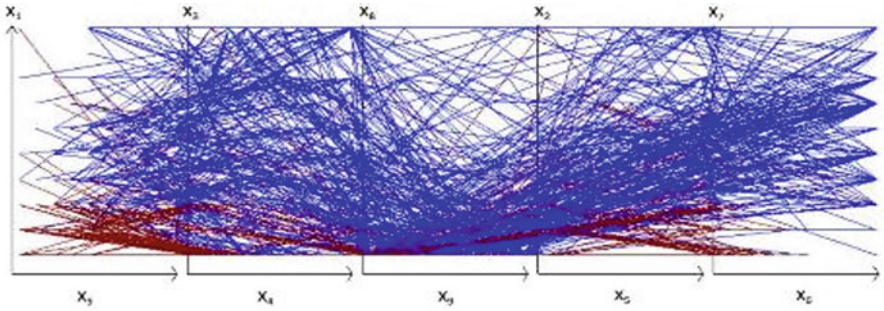


Fig. 13 Benign and malignant WBC data visualized in SPC as 2-D graphs of 10-D points

x_n) in (X_{n-1}, X_n) to form a directed graph x^*. Figure 12 shows the same 6-D point visualized in SPC in two different ways due to different pairing of coordinates.

The FSP algorithm has three major steps: *Filtering* out the less efficient visualizations from the multiple SPC visualizations, *Searching* for sequences of paired coordinates that are more efficient for classification model discovery, and *Presenting* the model discovered with a best SPC sequence, to the analyst [26]. The results of FSP applied to CPC graphs of WBC data are shown in Figs. 13, 14, and 15. Figure 13 illustrates the motivation for filtering and searching in FSP. It shows WBC data in SPC, where graphs occlude each other making it difficult to discover the pattern visually. Figure 14 presents the results of automatic filtering and searching by FSP algorithm. It shows only cases that are located outside of a small violet rectangle at the bottom in the middle, and go inside of two larger rectangles on the left. These cases are dominantly cases of the blue class. Figure 15 presents the remaining cases, which go through the small rectangle and do not go to the larger ones. The most of these cases are from the red class. Together these properties provide a rule:

$$\text{If } (x_8, x_9) \in R_1 \,\&\, (x_6, x_7) \notin R_2 \,\&\, (x_6, x_7) \notin R_3 \text{ then } \mathbf{x} \in \text{class Red else } \mathbf{x} \in \text{class Blue},$$

where R_1 and R_2 and R_3 are three rectangles described above. This rule has accuracy of 93.60% on all WBC data [26]. This fully interpretable rule is visual and intelligible by domain experts, because it uses only original domain features and relations.

This case study shows the benefits of combining analytical and visual means for interpretable knowledge discovery. The analytical FSP algorithm works on the multiple visual lossless SPC representations of n-D data, to find the interpretable patterns. While occlusion blocks discovering these properties by visual means, the analytical FSP algorithm discovers them in the SPC simplifying the pattern discovery, providing the explainable visual rules, and decreasing the cognitive load.

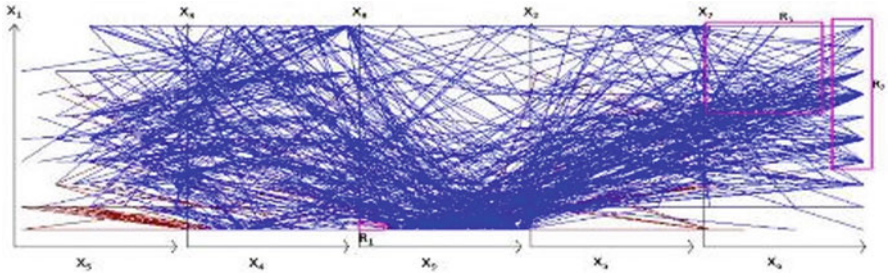


Fig. 14 SPC visualization of WBC data with areas dominated by blue class

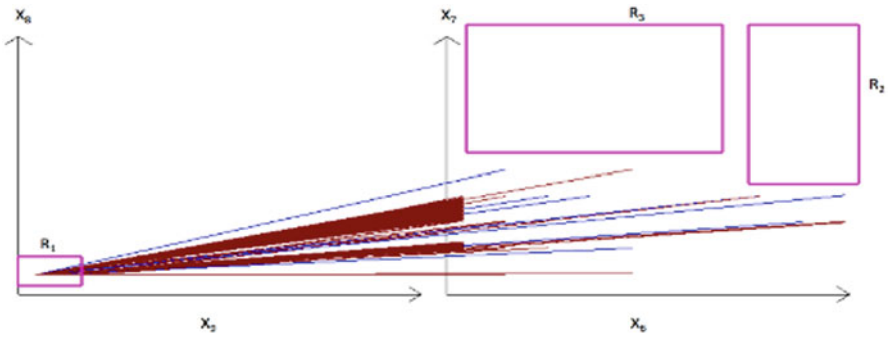


Fig. 15 WBC data in 4-D SPC as graphs in coordinates (X_9, X_8) and (X_6, X_7) that go through rectangle R_1 and not go to rectangles R_2 and R_3 in these coordinates

4.3 Case Study with GLC-L+CNN Algorithm

This case study uses the same WBC data as above. The occlusion was a major challenge there, which was dealt by FSP algorithm. Below for this, we use a combination of GLC-L algorithm, described in Sect. 4.1, and a Convolutional Neural Network (CNN) algorithm. The first step is converting non-image WBC data to images by GLC-L and the second one is discovering a classification model on these images by CNN. Each image represents a single WBC data case as a single polyline (graph) completely avoiding the occlusion. Figure 16 illustrates this design. It resulted in 97.22% accuracy on tenfold cross-validation [7].

4.4 Case Study with CPC-R+CNN Algorithm

This case study uses the same WBC data as the case studies above, but with CPC-R algorithm [21] for converting non-image data to images, and CNN algorithms for

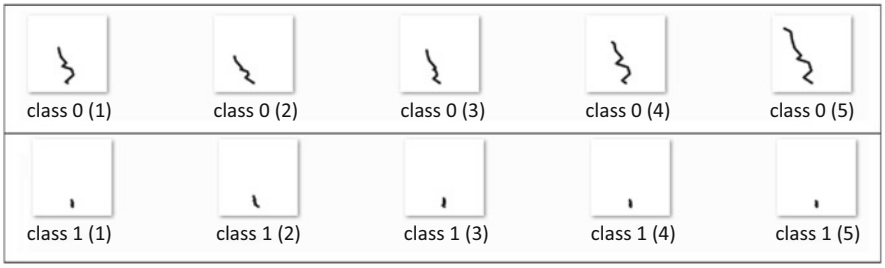


Fig. 16 WBC data samples visualized in GLC-L for CNN model

discovering the classification model in these images. Each image represents a single WBC data case, as a set of squares with a different level of intensities and colors.

The CPC-R algorithm is a modification of Collocated Paired Coordinates (CPC) algorithm [22]. The CPC algorithm first splits attributes of an n-D point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ to consecutive pairs $(x_1, x_2), (x_3, x_4), \ldots (x_{n-1}, x_n)$. If n is an odd number, then the last attribute is repeated to get $n + 1$ attributes. Then all pairs are shown as 2-D points in the same 2-D Cartesian coordinates and connected by arrows to form a directed graph $\mathbf{x}^*$: $(x_1, x_2) \rightarrow (x_3, x_4) \rightarrow \ldots \rightarrow (x_{n-1}, x_n)$. This graph is equivalent to the n-D point $\mathbf{x}$ and it can be *fully* restored from the graph.

The CPC-R algorithm, instead of connecting pairs (x_1, x_2) by arrows, uses the grayscale intensity from black for (x_1, x_2) and very light gray for (x_{n-1}, x_n) for cells. Alternatively, intensity of a color is used. This order of intensities allows full restoration of the order of the pairs from the image. The size of the cells can be varied from a single pixel to dozens of pixels. For instance, if each attribute has 10 different values then a small image with 10×10 pixels can represent 10-D point by locating five gray scale pixels in this image. This visualization is lossless when values of all pairs (x_i, x_{i+1}) are different and do not repeat. An algorithm for treatment of colliding pairs is presented in [21].

Figure 17a shows the basic CPC-R image design and Fig. 17b shows a more complex design of images, where a colored CPC-R visualization of a case is superimposed with mean images of the two classes, which are put side by side, creating double images. The experiments with such images produce accuracy between 97.36% and 97.80% in tenfold cross-validation for different CNN architectures on benchmark datasets [21]. The advantage of CPC-R is in lossless visualization of n-D cases, and the ability to overlay them using heatmap with salient points discovered by the CNN model, for model explanation. See Sect. 6 for more details on heatmap explanations.

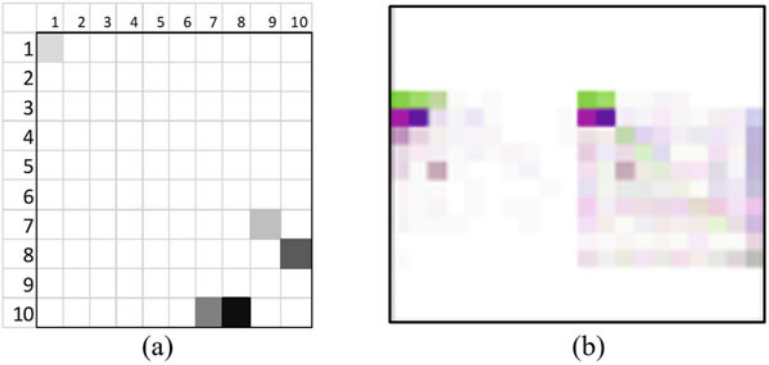


Fig. 17 CPC-R visualization of non-image 10-D points. (**a**) 10-D point (8, 10, 10, 8, 7, 10, 9, 7, 1, 1) in CPC-R. (**b**) Visualization in colored CPC-R of a case superimposed with mean images of two classes put side by side

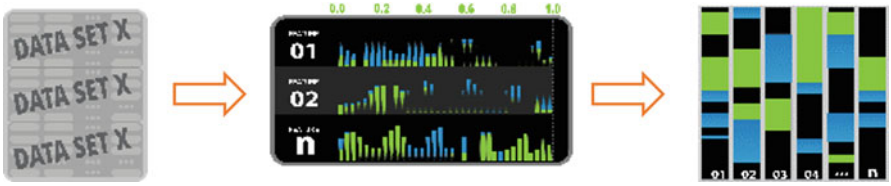


Fig. 18 DCP algorithm process

4.5 Case Study with RPPR Algorithm

This case study continues using the same WBC data as case studies above, but using Reverse Prediction Pattern Recognition (RPPR) algorithm [37]. This algorithm incorporates the prior DCP algorithm [20]. The goal is reaching both interpretability and high accuracy with of RPPR at the level or above it for non-interpretable algorithms on the same WBC data. The DCP algorithm starts with producing the dominance classifier structure $S = \langle\{V_i, h_{1i}, h_{2i}, \ldots, h_{ki},\}\rangle$, essentially a table containing intervals $\{V_i\}$ and the number of cases $h_{1i}, h_{2i}, \ldots, h_{ki}$, of each class on the training data within the respective interval on each predictor attribute X_i.

The next steps are combining dominance intervals in the voting methods, learning parameters of dominance intervals and voting methods for prediction, visualizing the dominance structure, and explaining the prediction. Figure 18 illustrates this process of construction of dominance intervals. The RPPR algorithm boosts DCP by discovering pair relations between attributes, then learning from said relations to override inaccurate DCP predictions. Figure 19 shows the result of the DCP algorithm on WBC data, with a benign case on the left and malignant case on the right, where each column represents an attribute, with colored intervals being the dominant intervals for red and blue classes, respectively.

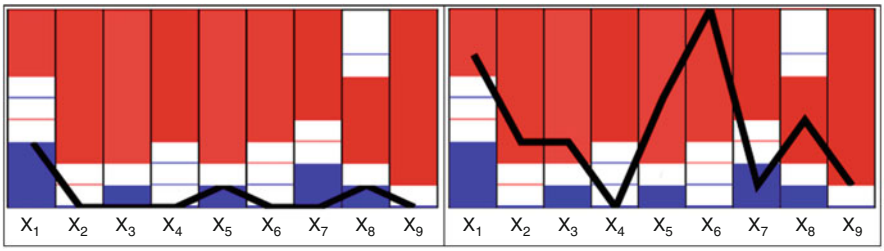


Fig. 19 Visualization of WBC data dominant intervals with a benign (on the left) and a malignant case (on the right)

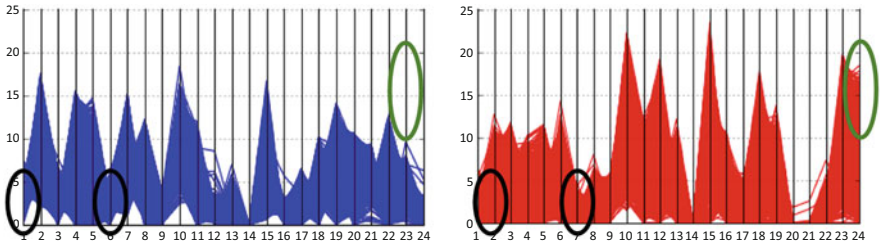
The steps of RPPR algorithm are: (1) *Encoding* elements of DCP algorithm as Boolean vectors, (2) *Finding* training cases *misclassified* by the DCP, (3) *Discovering* all the *unique* pairs for DCP False-Negative (FN) and DCP False-Positive (FP) *n*-D points on training data, (4) *Finding* FN and FP *n*-D points in the *validation/test*ing dataset with these unique pairs, and (5) *Reversing* prediction for these *n*-D points. Boosting DCP with RPPR allowed achieving accuracy over 99%, reaching the accuracy of non-interpretable algorithms and beyond using tenfold cross-validation on this WBC data. For more details and other experiments, see [37]. As far as we know, for this benchmark dataset, for the first time it is possible to have both accuracy and interpretability, rather than having to choose one at the expense of the other.

5 Scaling of Visual Discovery of ML Models

This section presents methods for scaling GLC-based methods of visual knowledge discovery and machine learning to large structured and unstructured data that is important for many application domains.

5.1 GLC and Embeddings to Cut Dimensions

The combining methods of Visual Knowledge Discovery (VKD) with Dimension Reduction (DR) methods expands applicability of VKD to large high-dimensional data. The approaches include combining General Line Coordinates (GLC) with different embeddings. Embedding [9] as mapping from discrete objects of very different nature, such as words, images, and graphs to vectors of real numbers have been very productive in many ML and DNN applications. However, usually new dimensions lack interpretability and special efforts are needed to interpret them. The combination consists of two steps. The first step is applying embedding to get



(a) (b)

Fig. 20 Comparing encoded digit 0 and digit 1 on the parallel coordinates using 24 dimensions found by the Autoencoder among 484 dimensions. Each vertical line is one of the 24 dimensions. (**a**) Digit "1". (**b**) Digit "0"

data of the lower dimension k from original n-D data and the second step is using lossless GLC-based algorithms, like used in cases studies above, on k-D data for ML model discovery. Below we illustrate this approach on classification of MNIST images of handwritten digits "0" and "1" [25].

At the *first step*, the Neural Network auto-encoder converts each 22×22 image (484-D point) to a vector of 24 features (24-D point). At the *second step*, these 24-D points are visualized in Parallel Line Coordinates. See Fig. 20 for digits "1" and "0". The *third step* is searching for discriminating features in these visualizations for the given digits. The direct observation of visualizations allows seeing the differences in these 24 coordinates that are shown by black and green circles. For instance, in Fig. 20, x_2 and x_7 are above the zero for all "1", but can be zero for "0". More specifically, $x_2 > 3$ and $x_7 > 2$ for all "1". Also, $x_{24} < 7$ for all "1", but it can be greater than 7 for "0". The *fourth step* is designing rules from discriminating features, directly from visualization without any computation, and explaining them. For instance, the observed properties of x_2, x_7 and x_{24} allow generating rules R_1–R_3:

$$R_1 : \text{if } x_2 < 3 \text{ then "0"}; \; R_2 : \text{if } x_2 < 2 \text{ then "0"}; \text{ if } x_{24} > 7 \text{ then "0"}.$$

These rules can be converted to a more conservative single rule:

$$R_4 : \text{if } (x_2 < 3) \,\&\, (x_2 < 2) \,\&\, (x_{24} > 7) \text{ then "0"}.$$

The *fifth step* is removing the cases, which satisfy these rules, and then searching for rules from the remaining cases, in the same visual way as in the above steps, or assisted by analytical ML methods. The *sixth* step is tracing these 24 features in the images of "0" and "1" to find their origin for interpreting them and to make the classification rules/models intelligible. See Sect. 6 such for methods. Many other DR methods, not only auto-encoders, can be used in this approach. For instance, the first k principal components of Principal Component Analysis (PCA) can be

used for DR as embedding, instead of the Auto-encoder. Similarly, it is not required to use the Parallel Line Coordinates, to visualize the reduced set of coordinates losslessly. Many other GLC- based methods can be used in this approach. The important advantage of this approach is the ability to *control the loss of original n-D information in the DR process*. For PCA, it can be done by selecting the first k principal components that cover, say, 90% of the total variance.

5.2 Dimension Reduction and Visual PCA Interpretation with GLC

While PCA is a most known and commonly used method for Dimension Reduction (DR) and n-D data visualization, it suffers from two major deficiencies: (1) difficulties to explain principal components and (2) loss of information when only the first two principal components are used to visualize n-D data.

The GLC-PCA algorithm presented below provides a solution for both challenges. It combines a visual explanation of each of the principal components using the GLC-Linear (GLC-L) visualization algorithm [22], and the one described above, which visualizes the n-D linear functions, in 2-D and Shifted Paired Coordinates [22], also described above, to visualize all the n principal components in 2-D, without the loss of information.

Figure 21 illustrates the GLC-PCA visualization algorithm, which allows representing all PCA principal components, without loss of information, using two types of General Line Coordinates (GLC) – Shifted Paired Coordinates (SPC) and GLC-L. In Fig. 21, the value of *explained variance (EV) ratio* for each principal component w_i is shown next to its name w_i. All principal components are ordered according to their EV value. The black polyline (graph) shows losslessly all 8 Principal Components of 8-D data, as a sequence of pairs in SPC: $(w_1, w_2) \rightarrow (w_3, w_4) \rightarrow (w_5, w_6) \rightarrow (w_7, w_8)$.

The red and blue polylines show, in GLC-L coordinates, how each Principal Component w_i is formed as a linear combination of the original attributes x_1–x_8 of the 8-D point x. The length of each red and blue segment shows the value of the respective original attribute x_j. The angle q_{ij} of the segment to the respective

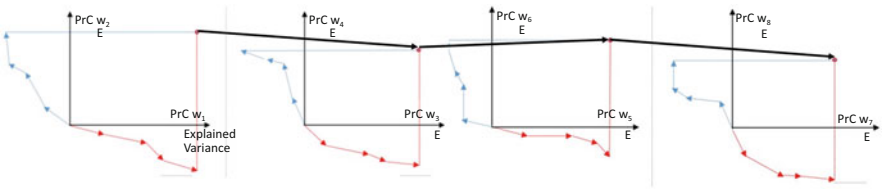


Fig. 21 All PCA principal components w_1–w_8, visualized in Shifted Paired Coordinates, and GLC-Linear coordinates, for an 8-D data point $x = (x_1, x_2, \ldots, x_8)$, without loss of information

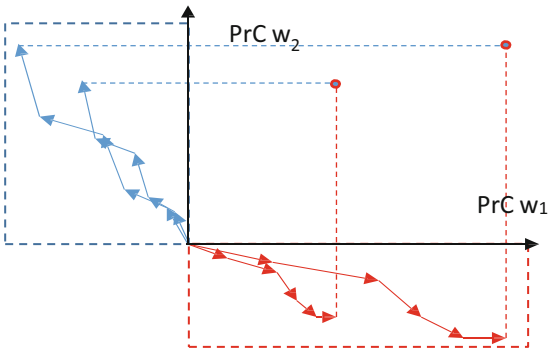


Fig. 22 Two 4-D points in the first two principal components in GLC-PCA

coordinate represents the contribution of the original attribute x_i to the principal component w_i. The original attributes x_j with smaller angles q_{ij} are most contributing to the principal component w_i. Figure 22 shows two 4-D points in the first two principal components in GLC-PCA.

The idea of GLC-PCA algorithm is visualizing not only the first two components y_1 and y_2 of each n-D point y, as it is done in visualization using PCA, but other y_i components of y too. In addition, the GLC-PCA algorithm shows visually how each y_i from y is formed as a linear combination of x_i of n-D point x. GLC-PCA allows the visualization of dimension reduction naturally. For instance, let the first 4 principal components out of 8 principal components, cover 85% of total variance and it is considered that these 85% are sufficient for the task then only these 4 will be visualized in GLC-PCA and used later for the model discovery.

5.3 Cutting the Number of Points and Dealing with Complex Data

Another aspect of visual knowledge discovery at scale is dealing with a large number of n-D points not only with large dimensions. A common approach to move from big data to smaller data is selecting a data subset, which captures properties of big data. The most popular idea here is data clustering with selecting representative cases from each cluster for further model discovery. Such clustering often is conducted by solving the optimization problems to maximize different similarity measures between n-D points. The review of these approaches can be found in [8]. This general idea is specified for different data types such as sequential and relational data, which is the area of active research [8, 34].

6 Visual Explanation of Analytical ML Models

Often visual ML model explanation relies on the same techniques as ML model visualization, while the goal of explanation is more specific. The ML model visualization typically does not produce an explanation itself but can create a basis for deriving the explanation. In this section, we focus on conceptual differences between the different methods of *visual explanations*.

6.1 Types of Explanations

While multiple definitions of terms "understanding", "interpreting", and "explaining" exist [1, 33] we favor one that requires describing the trained ML model in terms of domain ontology without using terms that are foreign to the domain where the ML task must be solved. Below we focus on externally and internally interpreted ML models.

The **externally interpreted models** are trained ML model explained in terms of interpretable input data and variables, but without interpreting the model structure. In [36] it is called as *post-hoc interpretability, functional understanding,* and *decision understanding*. This type of explanation is common for trained "black-box" Deep Neural Networks (DNN). Visualization of the space of input and output data of the trained ML model is an attractive approach for visual explanation of the trained ML model. It allows seeing the borders between classes produced by the model, e.g., see Figs. 2 and 5 in this chapter. While it is often challenging for the high-dimensional data, lossless or hybrid visualization methods make it feasible for many datasets.

The **internally interpreted models** are trained ML models explained in terms of interpreted elements of their structure not only inputs, e.g., trained decision trees, which are both internally and externally explainable ML models. Visualization of the model structure, in addition to visualization of the input data space and outputs of the model, is an attractive approach for this type of visual explanation.

Another aspect of the interpretability is its *coverage*: **entire model explaining** vs. **explaining individual model predictions**. Often individual predictions are explained using what we call *tabular pro-con explanation*. For a given new case **c** to be predicted, it shows pro cases (similar cases) and con cases (dissimilar cases) from classes. This is also an external explanation that does not go inside the model structure. This is a common explanation idea in *k*-nearest neighbors and case-based reasoning algorithms [33]. Respectively, there are two types of visual explanations based on (1) visualization of the entire model and entire input data space and (2) visualization of the given case and its nearest neighbors.

Two other explanation types are **explicit** and **implicit explanations**. Decision trees exemplify the former and heatmap-based explanations for deep neural network exemplify the latter. The next section is devoted to implicit explanations.

6.2 Heatmap Pixel-Based Implicit Explanations

Classification of an image A by deep neural network (DNN) can be **explained implicitly** by showing another image B, as an explanation. The idea is that if two inputs and predictions are similar then the explanation should be similar. If the similar property is not identified, then it is an *implicit explanation* that depends on the person who see the image B and needs to discover this similar property.

Often a similar image B contains **dominant (salient)** pixels of image A that are presented as an explanation of A. For example, DNN classifies the image A as a boat vs. a car and a truck [36]. The dominant pixels of image B represent the mast as a distinct feature of the boat relative to a car and a truck. Commonly such dominant pixels are colored according to their positive and negative scores for classes forming a **heatmap**.

While this is an acceptable *implicit explanation*, it assumes a human who recognizes a mast in these pixels. Thus, it is **incomplete explanation**, a more *complete explanation names a group of pixels*, e.g., mast. Next, this explanation is not applicable to another boat in the same image A, because that boat has no mast and requires its own explanation with another concept represented by its named group of pixels, e.g., people. Such conceptual explanations cannot be derived from the given trained DNN model, which recognized the boat, because it was not trained to recognize a mast or people explicitly. It is easy for a human to recognize such meaningful feature as a mast in a picture of the natural scene. In contrast, in medical imaging, if a radiologist cannot match DNN dominant pixels with the domain concepts such as tumor, these pixels will not serve as an explanation for the radiologist.

One of the methods to generate similar images B with dominant pixels is the *activation maximization*. The objective function of the optimization model for this maximization includes two components: one for maximization of activation and another one (regularizer) to ensure similarity of images [36]. The major challenges in activation maximization methods are (1) getting an image B with clear features, (2) selecting between competing images B (and dominant pixels) [38], and (3) interpreting dominant pixels as meaningful features.

Some alternative methods to find dominant (salient) pixels in DNN include: (i) *sensitivity analysis* by using partial derivatives of the activation function to find the max of its gradient, (ii) *Taylor decomposition* of the activation functions by using its first-order components to find scores for the pixels, (iii) *Layer-wise relevance propagation* (LPR) by mapping the activation value to the prior layers, and (iv) *blocking* (*occluding, perturbing*) sets of pixels and finding sets, which cause the largest change of activation value that can be accompanies by the class change of the image [36]. More approaches are reviewed and compared in [2, 5, 12, 40, 42, 47].

Two criteria are commonly used to find the sets of dominant (salient) pixels: (i) max contribution to output activation, and (ii) max of the changes in the output (class change). These criteria can contradict each other and, in general, *a difficult*

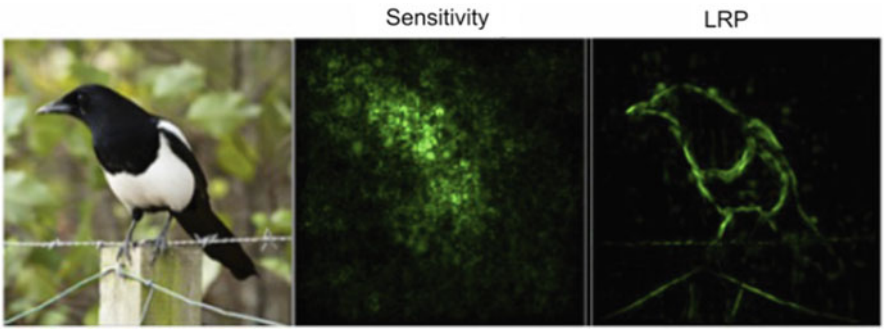


Fig. 23 Example of different heatmap methods [41]

multi-objective optimization problem must be formulated and solved. In practice, different linear combinations of them are used.

Guidotti et al. [13] review methods for explaining black box models based on Saliency Masks (maps), SM, visualization serving as meta-predictors that predicts the response of a black box to certain inputs. The sources of the SM are network activations. These approaches are known as Class Activation Mapping (CAM). They can visualize the linear combination of a late layer's activations, label-specific weights, or gradients by invoking back propagation and/or activation. Often these results are aesthetically pleasing providing heuristic explanations of image saliency [13]. For texts, it can provide a rationale (local "explanator) – a few words sufficient for the prediction of the original text.

Figure 23 illustrates the difference between the two heatmap methods. It shows that LRP explains the bird better, but this explanation is still implicit. The formal measure based on destroying some pixels and checking the change of the activation function captured this difference [41]. A human can understand that Fig. 23 shows a bird, due to the nose n, the tail t, and their relation $R(n, b, t)$ with the body b, which is between them. This *relation R* is not a part of the *heatmap implicit explanation*. A human derives it from the image, knowing that the body must be between the nose and the tail to be a bird. This is a common situation for all *heatmap implicit explanation* – they do not identify explicitly the *relations* between the features that they represent. This example shows the need to go beyond heatmaps.

7 Conclusion and Future Work

This chapter surveyed explainable machine learning approaches boosted by visual means. It includes motivation and comparison of analytical and visual ML methodologies, the input-based and structure-based types of methods of visualization of analytical ML. The chapter demonstrated that the approaches for discovering analytical ML models aided by visual methods are diverse and are growing. The

theoretical limits to preserve *n*-D distanced in lower dimensions are presented based on the Johnson-Lindenstrauss Lemma for point-to-point approaches. Further studies beyond the arbitrary points explored in this lemma are needed for the point-to-point approaches. In contrast, point-to-graph GLC approaches do not suffer from the limitations established in this lemma. Several real-world case studies, based on multiple GLC-based algorithms, had shown their advantages while multiple enhancements will be beneficial. The dimension reduction and clustering methods are outlined to support scalability and interpretability of the reviewed methods, including the visual PCA interpretation with GLC, and clustering for cutting the number of points.

Heatmap methods belong to the growing *sensitivity* [19] and the *attribution* [43] approaches that identify the elements of the input, which most affect the output. It is likely that heatmap implicit visual explanations will continue to be the focus of further studies, while this chapter has shown the need to go beyond heatmaps. Next, both the local and global explanation approaches need to be developed deeper. Below we list just a few future directions in interpretable visual knowledge discovery and ML among many others that can be developed. The first one is explaining ML models in human-relatable features in the Concept Activation Vectors (CAV) [19]. The next direction is combining neural networks with logic rules [15], which opens an opportunity to visual ML models, beyond the pixels and heatmaps, in line with the association rule visualization shown in Sect. 2.1. The GLC-based methods can contribute significantly to developing future interpretable ML models, due to their advantages such as lossless (reversible) visualization of *n*-D data, as graphs, and the interpretability of GLC-based predictive ML models.

References

1. Ahmad, M., Eckert, C, Teredesai, A., McKelvey, G. Interpretable Machine Learning in Healthcare, IEEE Intelligent Informatics Bulletin August 2018 Vol. 19, No. 1, 1–7.
2. Ancona M., Ceolini E., Oztireli A., Gross M., A unified view of gradient-based attribution methods for deep neural networks, *CoRR*, vol. abs/1711.06104, 2017. http://arxiv.org/abs/1711.06104.
3. Raschka S., MLxtend: Plotting Decision Regions, Journal of Open Source Software, 2018, https://doi.org/10.21105/joss.00638, https://rasbt.github.io/mlxtend/user_guide/plotting/plot_decision_regions/.
4. Cabrera A., Epperson WS., Hohman F., Kahng M., Morgenstern J., Chau D., FairVis: Visual Analytics for Discovering Intersectional Bias in Machine Learning, 2019, arXiv:1904.05419.
5. Choo J, Liu S. Visual analytics for explainable deep learning. IEEE computer graphics and applications. 2018 July 3;38(4):84–92.
6. Dasgupta, S.; Gupta, A., An elementary proof of a theorem of Johnson and Lindenstrauss, Random Structures & Algorithms, 22 (1): 60–65, 2003.
7. Dovhalets D., Kovalerchuk B., Vajda S., Andonie R., Deep Learning of 2-D Images Representing n-D Data in General Line Coordinates, Intern, Symp. on Affective Science and Engineering, pp. 1–6, 2018, https://doi.org/10.5057/isase.2018-C000025.
8. Elhamifar E., Recent Advances in Visual Data Summarization, CVPR 2019 Tutorial, https://rpand002.github.io/cvpr19_sumt.html.

9. Embeddings, Tensorflow guide, 2019, https://www.tensorflow.org/guide/embedding.
10. Facets visualizations for ML datasets. 2017. https://pair-code.github.io/what-if-tool/, 2018.
11. Fu C, Zhang Y, Cai D, Ren X. AtSNE: Efficient and Robust Visualization on GPU through Hierarchical Optimization. In: Proc. 25th ACM SIGKDD, 2019, 176–186, ACM.
12. Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter M, Kagal L. Explaining explanations: An overview of interpretability of machine learning. In: 2018 IEEE 5th Intern. Conf. on data science and advanced analytics (DSAA) 2018, 80–89, IEEE.
13. Guidotti R., Monreale A., Turini F., Pedreschi D., Giannotti F., A survey of methods for explaining black box models," *arXiv preprint arXiv:1802.01933*, 2018.
14. Hohman F, Kahng M., Pienta R., Chau D., Visual analytics in deep learning: An interrogative survey for the next frontiers. IEEE Vis. and Comp. Graphics, 25(8):2674–2693, 2019.
15. Hu Z, Ma X, Liu Z, Hovy E, Xing E. Harnessing deep neural networks with logic rules. arXiv preprint arXiv:1603.06318. 2016 Mar 21.
16. Inselberg, A., Parallel Coordinates, Springer, 2009.
17. Johnson, W., Lindenstrauss, J. Extensions of Lipschitz mappings into a Hilbert space. In Beals, et al. (eds) Conference in modern analysis and probability 1982. Contemporary Mathematics. 26. Providence, RI: AMS, 189–206, 1986.
18. Kahng M., Andrews P., Kalro A., Chau D. ActiVis: Visual exploration of industry-scale deep neural network models. IEEE Trans. on Vis. and Comp. Graphics, 24(1):88–97, 2018.
19. Kim B., Introduction to Interpretable Machine Learning Tutorial, CVPR 2018, http://deeplearning.csail.mit.edu/slide_cvpr2018/been_cvpr18tutorial.pdf.
20. Kovalerchuk B., Neuhaus, N. Toward Efficient Automation of Interpretable Machine Learning. In: Intern. Conf. on Big Data, 4933–4940, 978-1-5386-5035-6/18, 2018 IEEE.
21. Kovalerchuk B, Kalla DC, Agarwal B.: Deep Learning Image Recognition for Non-images. In: Integrating Artificial Intelligence and Visualization for Visual Knowledge Discovery 2022, 63–100, Springer, Cham.
22. Kovalerchuk, B. Visual Knowledge Discovery and Machine learning, 2018, Springer.
23. Kovalerchuk B, Grishin V. Reversible Data Visualization to Support Machine Learning. In: Intern. Conf. on Human Interface and the Management of Information 2018, 45–59. Springer.
24. Kovalerchuk B., Gharawi A., Decreasing Occlusion and Increasing Explanation in Interactive Visual Knowledge Discovery, In: Human Interface and the Management of Information. Interaction, Visualization, and Analytics, 505–526, 2018, Springer.
25. Kovalerchuk, B., Dovhalets, D., Constructing Interactive Visual Classification, Clustering and Dimension Reduction Models for n-D Data, Informatics, 4(23), 2017, http://www.mdpi.com/2227-9709/4/3/23.
26. Kovalerchuk, B. Visualization of multidimensional data with collocated paired coordinates and general line coordinates. *Proc. SPIE* 2014, *9017*, https://doi.org/10.1117/12.2042427.
27. Kovalerchuk B. Quest for rigorous intelligent tutoring systems under uncertainty: Computing with Words and Images. In: IFSA/NAFIPS, 2013, 685–690, IEEE.
28. Kovalerchuk, B.; Delizy, F.; Riggs, L.; E. Vityaev, Visual Data Mining and Discovery with Binarized Vectors, in: Data Mining: Foundations and Intelligent Paradigms, 24: 135–156, 2012, Springer.
29. Kovalerchuk, B., Balinsky, A., Visual Data Mining and Discovery in Multivariate Data using Monotone n-D Structure, In: Knowledge Processing and Data Analysis, Wolff, K.E et al., (Eds.), 297–313. Springer.
30. Kovalerchuk B., Schwing J., (Eds). Visual and spatial analysis: advances in data mining, reasoning, and problem solving. 2005, Springer.
31. Kovalerchuk B., Vityaev, E., Data Mining in Finance: Advances in Relational and Hybrid Methods, 2000, Kluwer/Springer.
32. Krause J., Perer A., Bertini E., A user study on the effect of aggregating explanations for interpreting machine-learning models. ACM KDD Workshop on Interactive Data Exploration and Analytics, 2018.
33. Lipton Z. The Mythos of Model Interpretability, Commun. of the ACM, 2018, 61, 36–43.

34. Liu S, Jampani V., Wang X, Batra D., Gupta A, Kautz J., Yang M-H, CVPR 2019 Tutorial on Learning Representations via Graph-structured Networks https://xiaolonw.github.io/graphnn/.
35. Maszczyk A., W. Duch, Support Vector Machines for visualization and dimensionality reduction, LNCS, Vol. 5163, 346–356, 2008, Springer.
36. Montavon G, Samek W, Müller KR. Methods for interpreting and understanding deep neural networks. Digital Signal Processing. 2018 Feb 1;73:1–5.
37. Neuhaus, N., Kovalerchuk, B., Interpretable Machine Learning with Boosting by Boolean Algorithm, Joint 2019 Intern. Conf. ICIEV/IVPR, Spokane, WA, 2019, 307–311. IEEE.
38. Nguyen A., Yosinski, J. Clune J, Multifaceted feature visualization: uncovering the different types of features learned by each neuron in deep neural networks, CoRR, arXiv:1602.03616, 2016.
39. Patel K., Bancroft N., Drucker S., Fogarty J., Ko A., Landay J., Gestalt: integrated support for implementation and analysis in machine learning. In Proceedings of the 23nd annual ACM symposium on User interface software and technology, 37–46. ACM, 2010.
40. Ribeiro M., Singh S., Guestrin C., Why Should I Trust You?: Explaining the Predictions of Any Classifier, Proc. the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 1135–1144, 2016.
41. Samek W., Montavon G., Müller K-R., Interpreting and Explaining Deep Models in Computer Vision CVPR 2018 Tutorial, http://interpretable-ml.org/cvpr2018tutorial/.
42. Samek W, Wiegand T, Müller KR. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296. 2017.
43. Sundararajan M, Taly A, Yan Q. Axiomatic attribution for deep networks. In: Proceedings of the 34th International Conference on Machine Learning-Vol. 70 2017, 3319–3328.
44. van der Maaten L. Dos and Dont's of using t-SNE to Understand Vision Models, CVPR 2018 Tutorial on Interpretable Machine Learning for Computer Vision, http://deeplearning.csail.mit.edu/slide_cvpr2018/laurens_cvpr18tutorial.pdf.
45. Wongsuphasawat K, Smilkov D, et al., Visualizing dataflow graphs of deep learning models in tensorflow. IEEE trans. on visualization and computer graphics. 2018: 24(1):1–2.
46. Zhang C. et al. Association rule based approach to reducing visual clutter in parallel sets, Visual Informatics 3, 2019, 48–57.
47. Zhang Q.-S., Zhu S.-C., Visual interpretability for deep learning: a survey, Frontiers of Information Technology & Electronic Engineering, vol. 19, no. 1, 27–39, 2018.
48. Dua, D. and Graff, C. Machine Learning Repository: Wisconsin Breast Cancer Dataset, Irvine, CA: University of California, https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original), 2019.
49. Kovalerchuk, B., Ahmad, M.A., Teredesai A., Survey of Explainable Machine Learning with Visual and Granular Methods beyond Quasi-explanations, In: Interpretable Artificial Intelligence: A Perspective of Granular Computing (Eds. W. Pedrycz, S.M.Chen), Springer, 2021, 217–267.

Visual Analytics and Human Involvement in Machine Learning

Salomon Eisler and Joachim Meyer

1 Introduction

The use of computers and sensors in practically all parts of life dramatically increased the amount of available data. Numerous visualization techniques and graphic tools have been developed to help people understand and analyze the information in the data. Keim [35] and others suggested that for data mining (an old synonym of data science) to be effective, humans have to be included in the data exploration process to combine the flexibility, creativity, learning capability, and general knowledge of the human with the enormous storage capacity and the computational power of today's computers. However, humans' cognitive abilities have not changed, creating a large discrepancy between the complexity of the data and human cognitive capacities. Since humans inspect the data largely through visualizations, the discrepancy can be seen as a problem of visual scalability, which is defined as the capability of visualization tools to effectively display large data sets in terms of either the number or the dimension of individual data elements [34, 41].

But what happens when humans do not have to "see" the information, when the analysis and learning processes are done by a machine? The current huge demand for data scientists indicates that even though ML and AI have become major tools for knowledge discovery with databases (KDDs) [23], humans are still strongly involved in the process. To this end, humans need to gain knowledge about data properties and the results of analytical procedures. These "insights" rely to a large extent on the visual display of information, i.e., visualization. The choice of the visualization method and its implementation will depend on properties of the data, the problem for which the data are analyzed, the purpose of the analysis, and other

S. Eisler · J. Meyer (✉)
Tel Aviv University, Tel Aviv, Israel
e-mail: jmeyer@tau.ac.il

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_40

factors [60, 65]. In the following sections, we will review the major tasks carried out by data scientists and the user interfaces and visual analytics related to them.

2 Overview: Visualizations Used During the Steps of the Machine Learning Process

In this section, we provide a high-level overview of the steps of machine learning (ML) and discern when and why visualizations can be used to improve the process. In the next section, we provide details for each step.

The seven steps of machine learning (ML) are data collection, data preparation, model selection, model training, evaluation and interpretation, parameter tuning, and prediction making [59, 28, 14]. They are similar to the process described by Fayyad for KDD [23]. Figure 1 depicts these seven steps of machine learning.

The analytical methods of ML usually require the data to have a certain form and structure. The process of converting the data into the required structure is called data preparation. Transforming the data to a tabular format, removing or inferring missing values, removing outliers and anomalies, and converting data to different types are examples of data preparation. Sometimes the techniques use categorical data, while others handle only numeric values. Usually, also, numerical values need to be normalized or scaled so that they are comparable [24]. During this step, several tasks often involve data visualizations to help detect relevant relations between variables or class imbalances, identify anomalies and outliers, and perform other exploratory analyses [59]. Visualizations for this step should not be very different from the visualizations used in classical KDD, with the exception that they often involve large amounts of data, and the data scientist can encounter visual scalability issues.

Model selection is determined by the business or scientific question that has to be answered using ML, the type of data, and its behavior. Often more than one model can answer the question. One therefore needs to understand the data to select the ML model, and visualizations may support this task. Again, as mentioned above, visualizations should not be very different from visualizations in traditional KDD. Even if visualizations are not used during model selection, it is important to review this step, because the selected model may also determine the visualizations used in the next steps of the process.

The model training step is when the model learns by using the data and computes its internal parameters. This is done on a set of data that should be as ergodic as possible, so that when the model is applied to the testing data and to real-world data, it will continue to provide good results. Here, visualizations are important, as they can help to monitor how the model converges to a solution.

The goal of the evaluation step is to assess the ML results rigorously to gain confidence that they are valid and reliable and that the model satisfies the original business goals before moving on [24]. This is done, using a separate set of data,

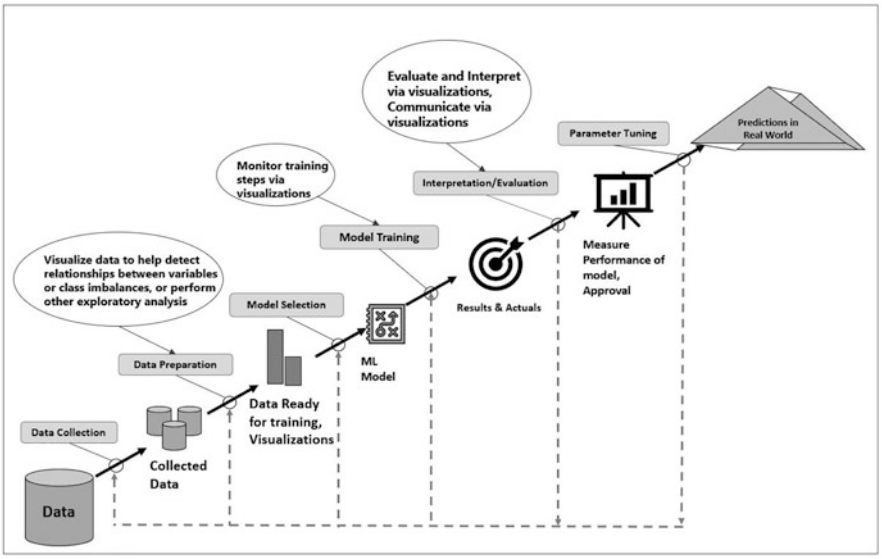


Fig. 1 The 7 steps of machine learning

called test data, with the expectation of getting results, similar to those achieved during the training step. Visualizations can also support this step very efficiently.

The interpretation step involves qualitative assessments. Various stakeholders have interests in the decision-making that will be accomplished or supported by the resultant models [24]. During this step, it is important to consider the comprehensibility of the model to stakeholders [24], and here visual analytics may be crucial.

Once the model has been defined, the model building is done with a programming language, supported by ML frameworks. More details will be introduced in Sect. 5.

The parameter tuning step refers to hyperparameter tuning, which is more an art than a technique. The objective is to improve performance through fine-tuning of the number of training steps, learning rate, initialization values and distribution, etc. [28, 59]. The values of hyperparameters configure characteristics of the model and may highly impact the training performance. However, given the complexity of the model algorithms and the training processes, identifying a sweet spot in the hyperparameter space for a specific problem can be challenging [39]. Dashboards of SPLOM, heatmaps, and line plots can be used to determine the optimal set of parameters.

The prediction in the real-world step uses data that were not used during the training to generate predictions [28, 59]. Prediction, or inference, is the step that provides the answers to the initial questions [28]. In this step, visualizations are mainly used for monitoring the results over time and to detect if the ML model needs to be modified because of changes in the data behavior. This is less relevant for the data scientist, and it is out of scope of this chapter.

In the next sections, we will discuss the visualizations for data preparation, model selection, training the model, evaluation and interpretation, and parameter tuning in more detail.

3 Visualizations in the Steps of the ML Process

3.1 Data Preparation

As mentioned above, during data preparation, one often wants to visualize data to help detect relevant relationships between variables or class imbalances, extract anomalies and outliers, and perform other exploratory analyses [59]. With the huge increase in the number of available observations and the number of features (variables) for each observation, it has become harder to provide clear and easy-to-understand visualizations. One way to overcome the visualization scalability problem is to use automatic analysis methods to extract potentially relevant visual structures from a set of candidate visualizations and to rank the visualizations in accordance with a specified user task [80]. Once a manageable number of potentially useful visualizations are identified, the data scientist can start interactive data analyses [80].

From a data perspective, the most popular automatized available tool for simplifying visualizations and for providing a better understanding is dimensionality reduction (DR). DR has become a core building block in the visualization of multidimensional data [74]. Examples of DR algorithms can be found in [21, 32]. Studies, [21, 16], have used scatterplots to evaluate users' perceptions for different DR algorithms on four different data sets and four techniques to achieve DR. Not surprisingly, performance depends on data characteristics [21], and the density and surrounding information affects the perception of clusters [76]. Performance will also be affected by characteristics of the individual user's cognition and perception [3, 27].

It is also possible to apply DR to network data that are visualized through graphs. A first step is to convert the graph data into a sparse matrix (called an adjacency matrix) that can be easily visualized. The second step is usually a matrix reorganization to reduce the graph to smaller graphs by partitioning its nodes into mutually exclusive groups [8]. The Laplacian matrix transformation is the most common example of this type of graph partitioning. The premise in the reordering of the adjacency matrix is to align the non-zero values close to the diagonal of the matrix, reducing the geometric distances between vertices, which results in a simpler visualization of graph partitioning [8]. Figure 2 shows graph visualization of 15,000 nodes from an Email network, based on traffic data collected for 112 days [18].

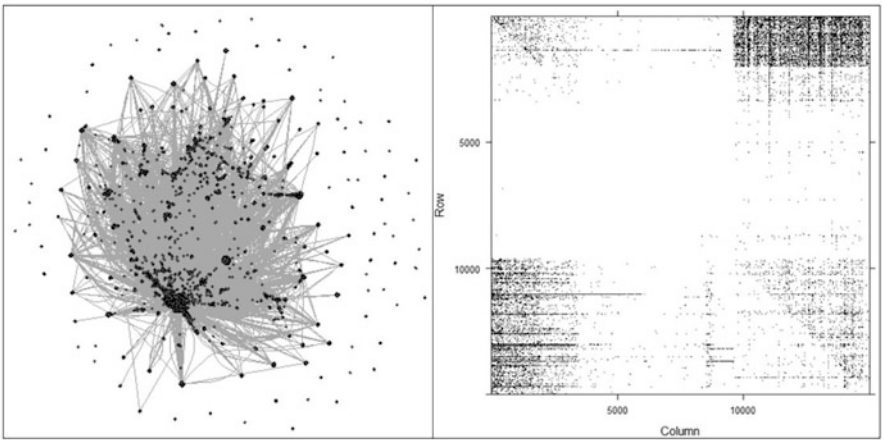


Fig. 2 Left, graph visualization; right, adjacency matrix using the R igraphs package

3.2 Choosing a Model

ML models are either unsupervised or supervised learning models, depending on the type of problems they are intended to solve. With supervised learning, the machine ("the learner") receives target information, along with a samples collection [24]. The machine must develop a solution, which will match the target with the sample data. In unsupervised learning, target information is not included. The machine is left to reach its own conclusions about the common properties that exist in the samples collection [24].

Numerous supervised and unsupervised ML methods exist and require different visualization techniques. It is beyond the scope of this chapter to present an exhaustive review of all types of visual analytics for all methods, and we will focus only on some popular ones.

The visualization techniques to be used for a specific machine learning model, as mentioned above, depend on the combination of the data types and the selected model [65]. The prevailing practice among data scientists is to use simple and intuitive visualizations: line plots, scatterplots, heat maps, contour plots, hierarchical trees, and several combinations of the above. The main differences between the visualization techniques are in the data elements that are visualized and the methods used to transform them.

On the other hand, when having to cope with very large numbers of features, these visualization techniques are less beneficial for data scientists. One method, which tries to overcome this high-dimensionality problem, is the use of parallel coordinates (also called PCP for parallel coordinates plot). PCPs were introduced by Inselberg [44, 45, 46, 47] as a different approach to visualizing multivariate data [50, 49]. Instead of graphing every pair of variables in two dimensions, the data are plotted repeatedly on parallel axes, and then the corresponding points are connected

with lines. There are many visualizations that are related to PCP either by sharing the typical parallel layout of axes or by the mapping of data points to lines [51]. PCPs treat each dimension uniformly. Because of this, users can make comparisons among dimensions without distortion. Uniform treatment becomes critical as the number of dimensions increases, as it is difficult, if not impossible, to make comparisons when dimensions are mapped to different visuals (shape, color, etc.) [48]. Parallel coordinates were originally used to visualize multivariate data by mapping different sequence charts, but they may also be used to present time-series data by mapping the different time steps to the individual axis [52]. The choice of the coordinate system will largely determine the patterns the visualization shows, and it is therefore important to learn how to "read" it. For parallel coordinates, this is not a trivial task, making its use less intuitive. R and Python already have libraries that support PCPs, so this visualization technique is available for data scientists. Figure 3 shows a PCP for the Iris data set. With it, one can see easily that there are three categories in the data.

The big challenge in parallel coordinates is the clutter produced by many lines, which hide the patterns contained in the data [51]. Lines use much more "real-estate" than points, so that the mass of data appears much larger in parallel coordinates than in scatterplots.

In parallel coordinates, the order of the axes is important. Reordering the axes may reduce clutter by revealing patterns that might have been hidden before. There are methods to reorder the axes [53].

Other variations of non-orthogonal axis visualizations are flexible linked axes [54, 55, 56], where a set of scatterplot matrices are linked with PCPs for analyses of high-dimensional data. Users can draw and drag axes freely, which is useful for different applications. Star Coordinates [48] is another example, where coordinates are arranged on a circle, sharing the same origin at the center. They use simply points to represent data, treating each dimension uniformly, even though this leads to rather crude representations. The purpose of the visualization is to gain insights and not accurate numerical analyses. A more recent and promising approach, which generalizes parallel, radial, and Cartesian coordinates, is lossless general line coordinates (GLCs) [58, 57]. Experiments showed [57] that methods using GLC can have higher expressiveness, while decreasing clutter and occlusion and simplifying visual patterns. At the same time, one can preserve all the multi-dimensional data in two dimensions by adjusting the GLC for the given multi-dimensional data sets. Still, this approach is so far not widely used.

Different approaches are used for image processing and data graphs. In face recognition, the visualizations can be the arrays of face images. With network data, the visualizations are graph representations of the network, using different layouts of the data points (either in 2D or 3D). Sometimes symbols are used to consolidate several data points, with specific meanings assigned to the types of connections of the group of data points to reduce the visual clutter. Below is a short overview of the different ML models and their related visual analytics.

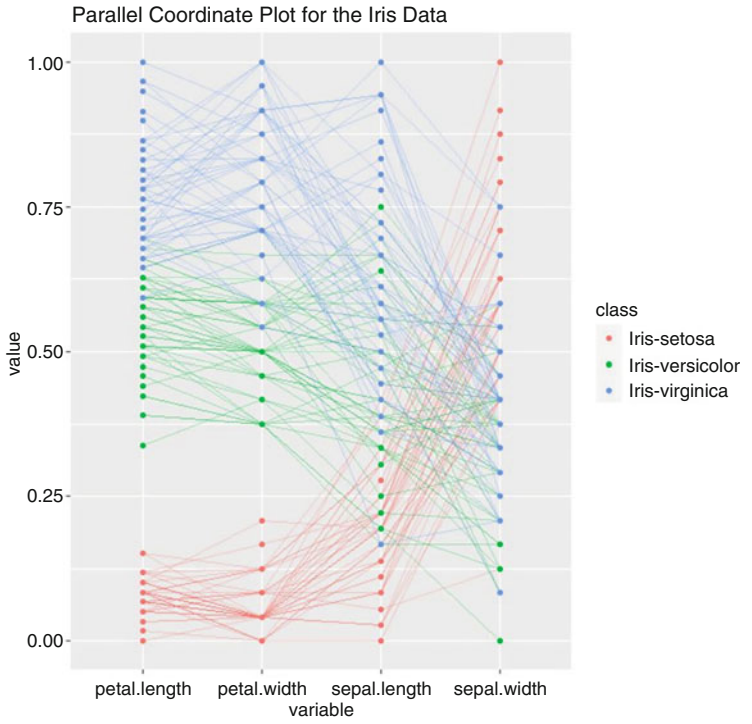


Fig. 3 Iris data set visualization with parallel coordinates, using Ggparcoord function from GGally R package

3.2.1 Supervised Models

Classification Trees Users either want to edit the tree (grow, prune, or optimize it), use the tree (classification), or analyze it (data exploration). Typically, users often switch between the edit and analysis process. For each task, one can identify important elements and extract requirements [82]. Several examples of classification tree models were cited in [32], and [82] present a good example of an interactive interface with a decision tree that supports editing, classifying, and exploring the tree.

Regression Models These models can be used to isolate the relationship between outcome and explanatory variables, while holding other variables constant. Visualizations are important when interacting with this kind of models, because it is possible to visually represent these relationships in an easy-to-understand way with simple scatterplots. When the relationship between an explanatory variable and the response depends on multiple regression coefficients, the model's fit is more readily understood with a visual representation than by looking at a table of regression coefficients [7]. Examples of classification regression models are shown

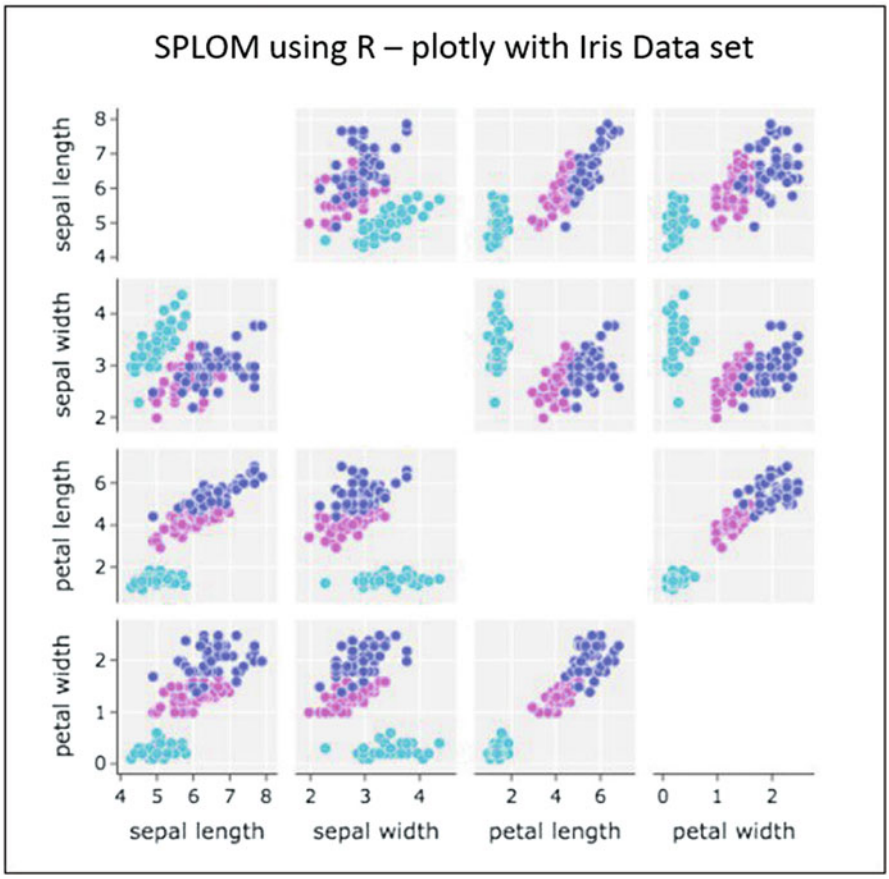


Fig. 4 SPLOM plot using R and the plotly package—[69]

in [32]. Figure 4 shows a scatterplot matrix (SPLOM) that can be used to quickly explore distributions (clustering—unsupervised) and relationships (regressions—supervised), based on the known Iris data set and created with R.

Cross-sectional plots, contour plots, and 3D representations of the regression surface [7] are helpful in visualizations for regression models. Figure 5 depicts examples of this type of visualizations.

Bayesian Networks (BNs) These plots, also known as belief networks [4], are related to the family of probabilistic graphical models (GMs). Their graphical structures are used to represent knowledge about an uncertain domain [4]. Visualizations of this type of models are similar to decision tree visualizations. An example of a tool that visualizes BNs is BayesViz's [13]. This tool presents the inferred network with edges, colored per correlation coefficient, and colormap tables, representing the conditional relationships between the values of parent and child nodes.

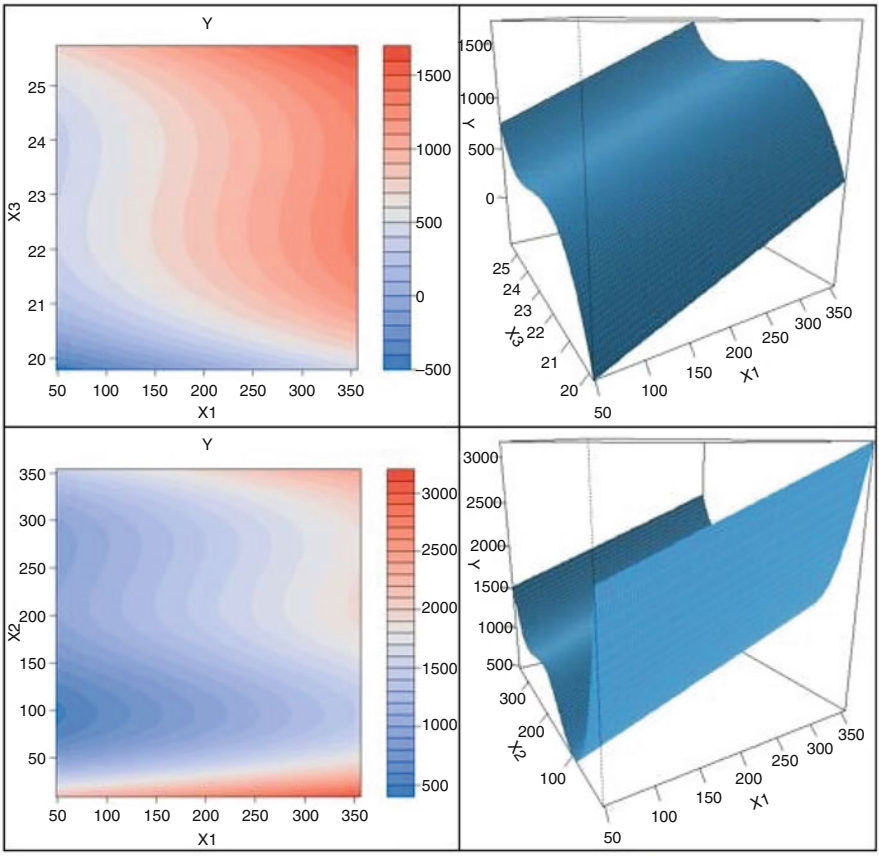


Fig. 5 Representations of the regression surface as a function of X1, X2, and X3, using synthesized data where Y = 22 + 3X1 + 2X2 + 3X3. Left: Filled contour plots. Right: Perspective plots. Using R and code from [7]

Decision Tree Inducers Decision trees are constructed with inducers (also called classifiers). A decision tree inducer is basically an algorithm that automatically constructs a decision tree from a given (training) data set. Visualizations here are practically the same as in the decision trees above. Several models of this type exist: ID3, C4.5, CART, CHAID, QUEST, CRUISE, and many others [32, 70].

Support Vector Machines (SVMs) These models are used for both classification and regression challenges. SVMs are the only linear models that can classify data that are not linearly separable [4]. Visualizations are used in SVM to understand the decision boundary in the space of input variables. This decision boundary is estimated from available training data but is intended to be used for classifying future input samples [13]. Examples of SVM models are shown in [32]. In [4, 13], it

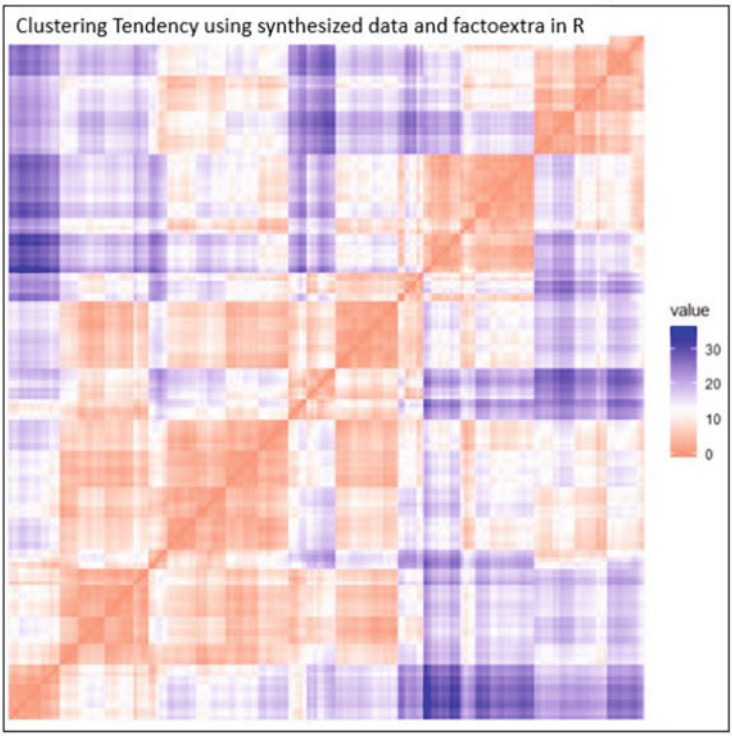


Fig. 6 Clustering tendency [33] is detected in a visual form by counting the number of square-shaped dark blocks along the diagonal in the image. The data set used is synthesized data shown in Fig. 7. Red: high similarity (i.e., low dissimilarity), Blue: low similarity

is possible to see a comparison of visualizations of the decision boundary for linear and nonlinear SVM models.

3.2.2 Unsupervised Models

Clustering Before applying any clustering algorithm to a data set, one first has to assess the clustering tendency (to understand whether the data set has natural clusters or not) (Fig. 6 [33]). The classification of objects into clusters requires methods for measuring the distance or the (dis)similarity between the objects. Visualizations can help to expose and analyze both the clustering tendency and similarity distances in the data [33]. Density-based clusters [20] are an additional way to visualize possible clusters. The main reason why it is possible to recognize clusters in Fig. 7 is that the density of points within each cluster is clearly higher than the density outside the cluster [20].

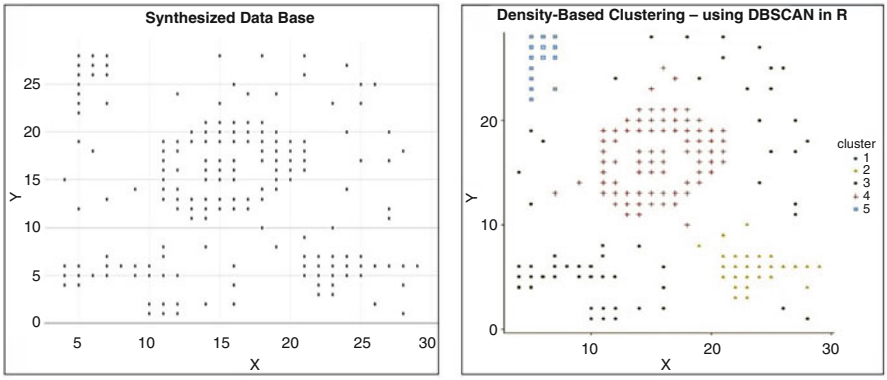


Fig. 7 Density-based clustering can help to find clusters with different shapes and sizes from data containing noise and outliers [33]

The percentage of data points above the similarity–dissimilarity line (PAS) shows the expected accuracy of the classifier, using a particular feature set [2]. The similarity–dissimilarity visualization for a high-dimensional feature space can provide very important information that can later help in the development of the models [2].

Hierarchical clustering is another type of a clustering analysis method that is well supported by visual analytics. Hierarchical clustering is an algorithm that is intended to create a hierarchy of clusters. The last layer of the hierarchy is a group of clusters, where each cluster is separate from the others, and the objects within each cluster are very similar to each other. Strategies for hierarchical clustering generally fall into two types: agglomerative and divisive. Agglomerative is a "bottom-up" method: each observation starts in its own cluster (leaf), and pairs of clusters are merged as one moves up the hierarchy until there is just one single big cluster (root) [33]. Divisive is a "top-down" approach: all observations start in one cluster (the root), and splits are performed recursively down the hierarchy.

The visual representation of hierarchical clustering is a tree-based visualization of the objects, which is also known as dendrogram [33]. It can also be used to analyze network data. Figure 8 depicts four visualizations of the same data set (Offences recorded by the police in England and Wales by offence and police force area for 2001/02, from https://www.gov.uk/government/statistics/historical-crime-data), with dendrograms and a heat map using R.

Graphs—Network Data A network consists of a collection of entities and the connections or relations between them. It can be visually represented as a graph, with vertices representing entities and edges representing their relationships [68]. This is an intuitive representation, because of the close similarity between the real world and the visualization. When data scientists explore the data via visualization, they almost explore the actual network. Networks are a special case, where the data scientist can use different ML techniques. These include routines for clustering,

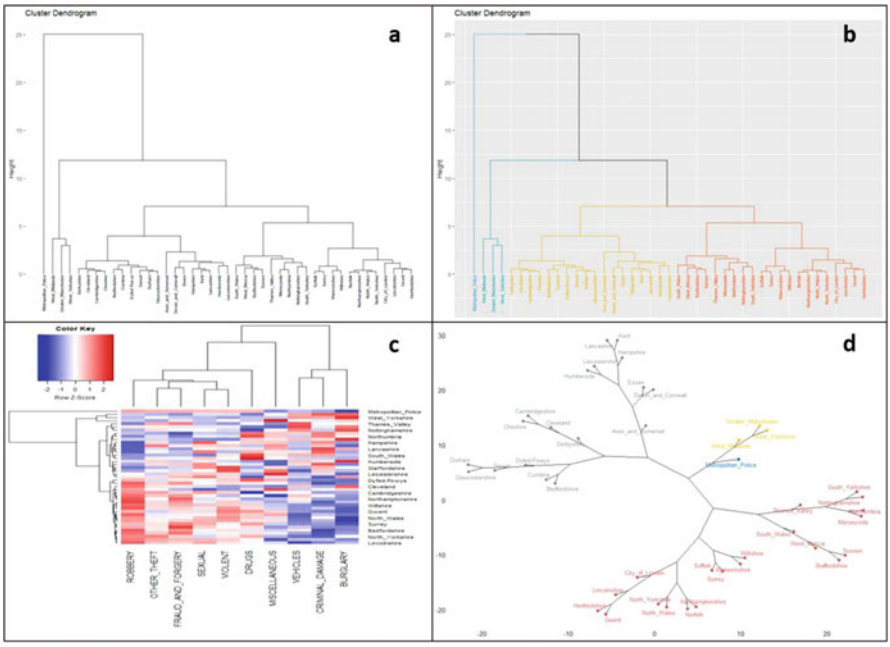


Fig. 8 (**a**) Simple dendrogram, (**b**) dendrogram with 4 groups, (**c**) heat map using package pheatmap, (**d**) phylogenic dendrogram. Visualizations were prepared with R

decomposition, random graph generation, statistical analysis, and calculation of network distances [12]. Graphs need a separate approach for almost all the 7 steps of the ML process. An example that specializes in graph data with specific graph visualizations is GRAPHVIS, which supports interactive techniques, such as brushing, linking, highlighting, as well as semantic zooming.

3.2.3 Advanced Methods

Deep Learning Deep learning includes a wide range of techniques, including neural learning networks, genetic algorithms [2], convolutional neural networks, recurrent neural networks, deep belief networks, etc. One characteristic of these methods is that it is hard to understand and interpret the underlying rules or mechanisms that produce the predictions, i.e., the methods are black boxes. These methods can be supervised or unsupervised. As we will see later, for the adoption and use of the model, it is often crucial to assure that one can interpret and explain the model, and this can be done using visualizations. One way of trying to interpret a neural network model is to create heat maps of the cells of the actual neural network. Reference [68] presents an example of a visualization for a neural machine that translates it. There, the x-axis and y-axis of a heat map plot correspond to the words

in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight ij of the annotation of the j-th source word for the i-th target word, which corresponds to the cells of the neural network.

Ensemble Models Ensemble models are models that make predictions, based on a group of different models. While deep learning models are more appropriate in fields, such as image recognition, speech recognition, and natural language processing, tree-based ensemble models frequently outperform standard deep learning models with structured data where features are individually meaningful [11]. Visualizations of these decision trees are relatively easy to understand, as they show a hierarchical view of the step decisions, made by the classification model.

3.3 Training the Model

The goal of training is to enable the machine to learn the data, so that the answers to questions or the prediction are as correct as possible. If the processing is too heavy and takes too long, independently from the ML framework used, trained models need to be saved in a file and afterward restored to compare the model with other models, to test the model on new data or for checkpoints. The saving of data is called serialization, and restoring the data is called deserialization. Supporting this task with visualizations can be very helpful to the data scientist. An example of a framework that enables this feature is TensorFlow with TensorBoard (its suite of visualization tools) [1, 6, 26].

One also often needs to monitor the algorithm iterations to verify convergence and to evaluate the results. Several types of visualizations enable the data scientist to manage this process. For example, the progress of gradient descent on a test surface can be visualized for all the steps [30]. Sometimes it is useful to display the three-dimensional data in two dimensions, using contours or color-coded regions [83]. Figure 9 depicts a contour plot for gradient descent. The red line shows the convergence of the process to the minimum of the cost function.

3.4 Evaluation and Interpretation

3.4.1 Evaluation

Human interaction is very important in the evaluation step of the process. The right data, combined with the right data science techniques, will help to identify the models that optimize a cost criterion. However, only humans can decide on what is the best criterion [24]. One strategy for making decisions is to rank a set of cases by scores and then take actions on the cases at the top of the ranked list. This can be achieved using profit curves, which consider costs/benefits related to

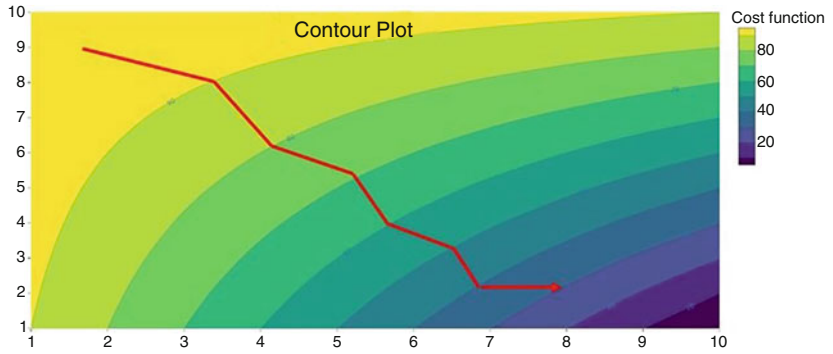


Fig. 9 Contour plot illustration using R. The red line shows the gradient descent convergence

true positives, false positives, true negatives, and false negatives. Profit curves are appropriate when the conditions under which a classifier will be used are known with high certainty. A profit curve can help optimize overall profit and help select the best model and predicted probability threshold [37]. Figure 10 shows an example of cumulative gains and profit curves for three classifiers, using R package modelplotr based on the Bank Marketing Data Set [64].

Different evaluation methods should be used when the conditions under which the classifiers are used are uncertain or unstable. One such method is the receiver operating characteristics (ROC) curve. ROC curves can serve as the basis for performance measurements in classification problems at various thresholds settings. ROC is a probability curve, and area under the curve (AUC) represents the degree of how much the model is capable of distinguishing between classes. The higher the AUC, the better the model predicts 0s as 0s and 1s as 1s or distinguishes between positives and not positives [66]. Figure 11 depicts an example of a ROC curve, and the diagonal line $x = y$ represents random performance, using the wine quality data set from https://archive.ics.uci.edu/ml/data-sets/wine+quality and the pROC R package, code from https://www.kaggle.com/milesh1/receiver-operating-characteristic-roc-curve-in-r. Another, more intuitive visualization for model evaluation is the "cumulative response curve." Cumulative response curves plot the true positive rate, which is the percentage of positives correctly classified, as a function of the percentage of the population that is targeted (x-axis) [24].

Another metric that can be useful for evaluating a clustering model is the silhouette coefficient (Si) [73]. It is a visualization method for interpreting and validating the consistency within clusters of data. A value of Si close to -1 indicates that the object is poorly clustered. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters. It thereby provides a way to visually assess parameters, such as the number of clusters. Figure 12 shows the silhouette plot of a k-means clustering, using the synthesized data set shown in Fig. 7.

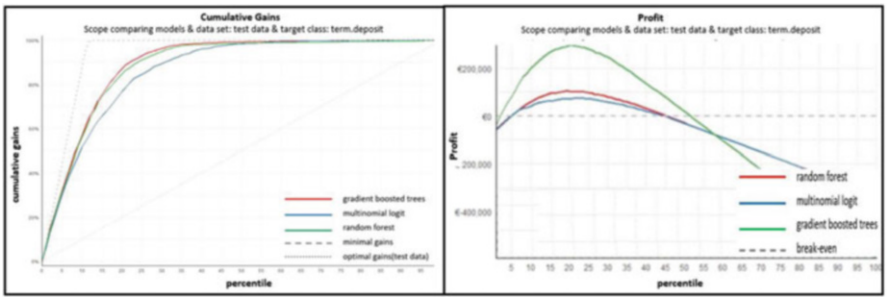


Fig. 10 Cumulative gains and profit plot for classifiers

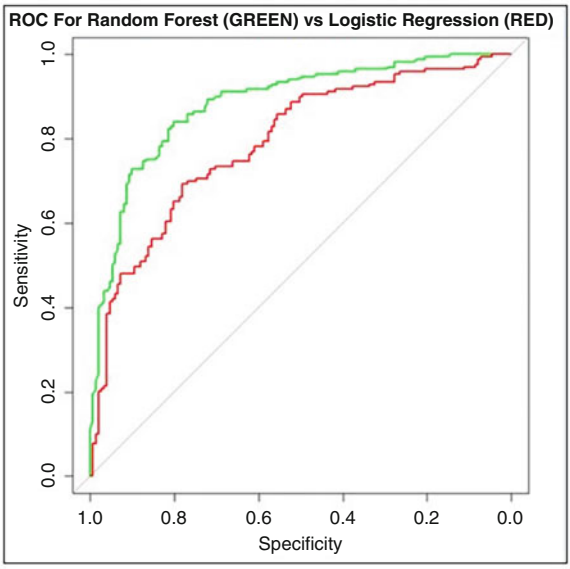


Fig. 11 ROC curve. The diagonal line $x = y$ represents random performance, comparing 2 classification models

Typically, data scientists will perform several training processes to be able to compare model performance. The visual comparison of model performance for several experiments can result in heavy cognitive load for the data scientist. A couple of examples that support this visual process are Squares [71] and iVisClassifier [15]. Squares has been used for displaying the performance of a classifier trained on a handwritten digits data set. iVisClassifier includes dimensionality reduction techniques, scatterplots, and parallel coordinates to support examination of model behavior.

In ML, a confusion matrix is commonly used to present the accuracy of a learning algorithm. It is possible to compare model performance, showing the evolution of the confusion matrix over the training steps [40]. Its visual representations are stacked plots that reveal the changes of true positives and false positives for each feature over iterations [40].

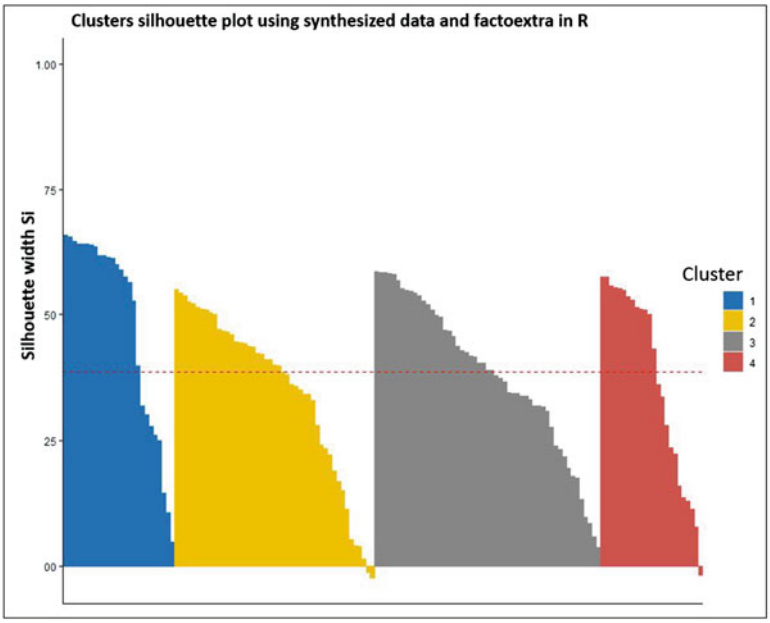


Fig. 12 [33] clusters silhouette plot, and silhouette width is also an estimate of the average distance between clusters. Values are between 1 and −1, with a value of 1 indicating a very good cluster

3.4.2 Interpretation

One of the most debated topics in deep learning is how to interpret and understand a trained model, particularly in the context of high-risk industries, such as healthcare [77]. The interpretation step includes interpreting the discovered patterns and possibly returning to any of the previous steps, as well as visualization of the extracted patterns, removing redundant or irrelevant patterns and translating the useful ones into terms users can understand [32]. The visualizations in this step are critical, as they can be crucial in getting the organization's trust to accept and adopt the specific ML model and algorithms [72]. Visualization methods provide the necessary means to simultaneously analyze the huge amount of information hidden in a deep learning neural network [5].

It is possible to typify ML models interpretability into two types [61] according to the models interpretability: glassbox and blackblox models. The first type is related to ML models that are interpretable by design, such as linear models, rule lists, decision trees, regressions, and generalized additive models [61]. The second type is related to ML models for which interpretability was not an intrinsic element of their design (such as CNNs) and that need an additional "ML" layer that uses explainability methods (i.e., partial dependence, LIME, SHAP) for explaining those models [61]. Here we will focus on the interpretability of blackbox ML models.

Interpretability Techniques for Blackbox Models Deep learning networks, such as neural networks, convolutional neural networks, or deep belief networks, are considered black boxes. It is possible to divide the visualization methods used to interpret the ML models into three classes: preliminary methods, qualitative patterns, network visualizations methods, and model-agnostic [62] quantitative methods.

Preliminary Methods [77] These are simple methods, which show the overall structure of a trained model. These methods just show a diagram of the neuron's connections. In [81], an example of such a method is shown for the overall structure of a model using the Theano framework.

Qualitative Patterns Network Visualizations Methods [19] These are qualitative representative methods that try to improve the CNNs interpretability using visually perceptible image patterns [19]. This approach is based on the CNNs hierarchical feature representation mechanisms that try to mimic (biomimicry) the hierarchical structure of the visual area in the human brain (this is also the reason why ML models for image processing are usually CNNs). With this type of methods, it is possible to elucidate the roles of individual neurons or groups of neurons and to gain insights on what they are doing [77].

To get insights regarding the association between features and neurons, the data scientist needs to seek the preferred stimulus to recognize the individual kind of the reaction and show the response to specific visual patterns. These methods tend to manipulate the gradients that are formed from a forward and backward pass while training a model. Saliency maps are a visualization technique based on gradients, to understand and visualize the nonlinearities embedded in feed-forward neural networks [63]. Examples can be seen in [79]. Color segmentation is used, because the saliency map might only capture the most discriminative part of an object, and saliency thresholding might not be able to highlight the whole object. Therefore, the map with the thresholding was propagated to other parts of the object, which was achieved using the color continuity cues [79]. A recent survey [19] identified four representative types for this approach: activation maximization, deconvolutional networks, network inversion, and network dissection.

Model-Agnostic Quantitative Methods Model-agnostic methods do not depend on the ML model to be interpreted and assign a number to each of the features that is related to its importance. Examples are partial dependence plots, accumulated local effects, feature interaction, feature importance, global surrogate models, prototypes and criticisms, local surrogate models (LIME), and Shapley values explanations and counterfactual explanations [62]. For a quite exhaustive review of those examples and more, see [62].

Two very known methods are Shapley Values and LIME. LIME tests what happens to the predictions when there are variations in the features data of the ML model, which is a very intuitive approach. However, the Shapley value, according to [62], might be the only method to deliver a full explanation for ML models, while

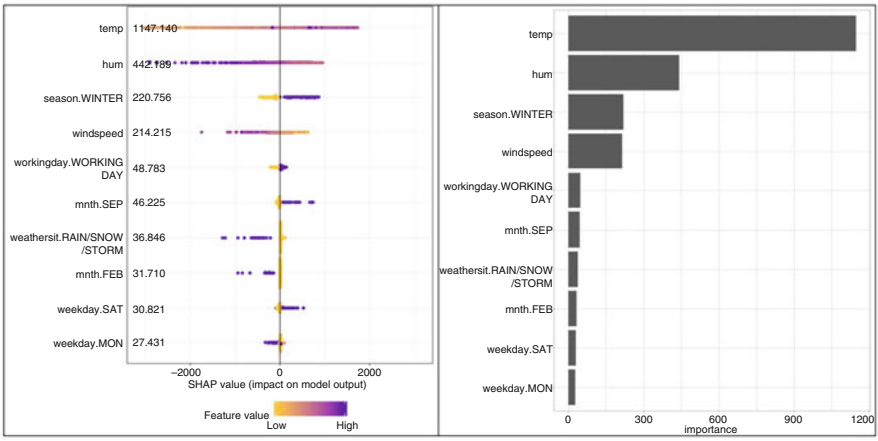


Fig. 13 SHAP plots for a rental bike data set [22] using R code from [9]. Left—SHAP summary plot of a 10-feature model. Right—importance plot, based on SHAP, using a classical bar chart, showing the importance of the predictors[42]

methods such as LIME only assume linear behavior of the ML locally that has not been theoretically proven.

The Shapley value has also some disadvantages [62] and may not be the preferred method for all cases. Its computation requires a lot of system resources. The Shapley value may not be efficient if one needs an interpretation that uses only a subset of the features because it always uses all the features. For selective explanations, LIME or a variation of the Shapley value method called SHAP (SHapley Additive exPlanations) [43, 42], which does not require the full features set [62], may be better choices. Another possible disadvantage of SHAP, compared to LIME, is that SHAP does not provide explanations regarding the "sensitivity" of the predictions to changes of the features (which exist in LIME) [62]. SHAP only provides the feature importance for every feature. The impact of features is frequently plotted with bar charts to represent global feature importance, or with a partial dependence plot to represent the effect of changing a single feature [25]. SHAP summary plots replace typical bar charts of global feature importance, and SHAP dependence plots provide an alternative to partial dependence plots that capture interaction effects better [42]. Figure 13 depicts a SHAP summary plot of a 10-feature model. The y-axis indicates the variable names, in order of importance from top to bottom. The value next to them is the mean SHAP value. On the x-axis is the Shapley value, which indicates the change in log-odds. From this number, it is possible to extract the probability of success. Gradient color indicates the original value for that variable. Each point represents a row from the original data set [10, 9].

Figure 14 depicts SHAP interaction dependence plots, which use the SHAP value of a feature for the y-axis and the value of the feature for the x-axis to present how the feature's attributed importance changes as its value varies. SHAP dependence plots capture vertical dispersion, due to interaction effects in the model. These

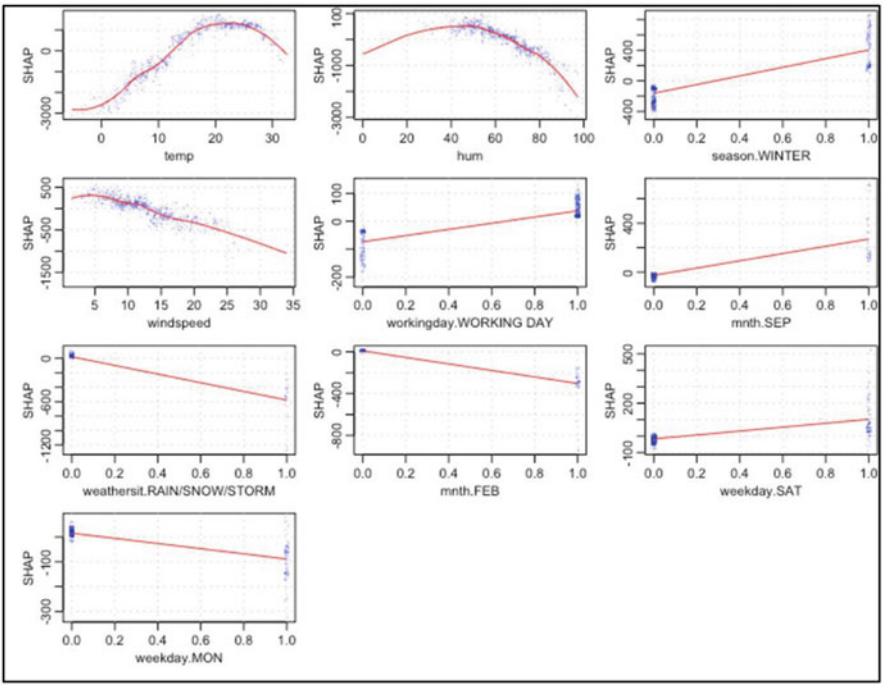


Fig. 14 SHAP dependence plots for each of the 10 features from the rental bike data set [22], using R code from [9]

effects can be visualized by coloring each dot with the value of an interacting feature.

As mentioned above, graph data need different treatments for almost all 7 steps of the ML process. There are two main entities that are usually considered when exploring graph data: Triplets and Motifs. Computing triangles in graphs has wide applications in network analysis, for identifying dense subgraphs, and for uncovering hidden thematic layers [8]. Triangle computations help to visualize clusters in graphs and support the data exploration during data preparation. Motifs are frequently recurring patterns of basic structural elements that occur in graphs [17]. They are small, local patterns of interconnections that occur throughout a network with significantly higher probability than in random networks. Motif detection is helpful for interpreting the graph. One way to detect and to simplify motifs visualizations is by replacing motifs in the network with easily understandable glyphs [17].

A similar approach, which goes beyond motifs, is identifying and visualizing clusters of motifs, called modules in [38], in the graph and then visualizing the relations between the modules in a hierarchical way [38]. This reveals graph patterns for each module and allows users to gain a better understanding of the structure of

the graph. ModulGraph [38] graph visualization can show the relations between the communities of motifs and the different kinds of communities.

Another element, when exploring a graph, is to use graphlet frequencies to analyze the topological similarity of its subgraphs, using graph kernels [36]. In simple terms, the graphlet frequency vector of a graph is like the feature vector of the graph [36]. If the graphlet frequency vectors are not similar, then the layout of each graph will look different.

3.4.3 Hyperparameters Tuning

Today's hyperparameter tuning processes are highly empirical, using rules-of-thumb. They are "human driven," as they are performed manually by the data scientists. The ML tuning step has been described as a "a project involving multiple experiments, which may last several hours or days." The number of hyperparameters differs between models, but there may often be more than 12 parameters. The optimizing algorithm, dropout rate, the number of layers, and the width of each layer are hyperparameters that are commonly tuned. For supervised models, the key performance metrics (KPIs) are usually accuracy, precision, recall, and ROC curves. Additional KPIs are the learning curve (how steep is the initial step and when does it approach the asymptote), training loss vs. validation loss (to detect overfitting) [39]. To create visual analytics for hyperparameter tuning, it is necessary to capture and efficiently store the received KPIs, together with the employed parameters, per each tuning round. In this regard, this is similar to the traditional BI approach, where the data are stored and then visualized, using dashboards with combinations of SPLOM, heatmaps, and line plots for different combinations of the KPIs and hyperparameters. Such an approach was described by Tian [39].

3.4.4 Summary

In Fig. 15, we summarize the visualization methods, using a mapping of the techniques related to 6 of the 7 machine learning steps: data preparation, training, evaluation, interpretation and hyperparameters tuning for classic supervised and unsupervised models, deep learning models, and ensemble models.

4 User Interfaces and Frameworks

As mentioned in section 2, once the model has been defined, it is necessary to build it. Model building is done, using a programming language, such as Python, R, C++, JavaScript, or Java (see [29, 67, 78] for a more complete list), supported by well-known ML frameworks such as TensorFlow, Torch, PyTorch, Jupyter Python, etc. (see [29, 67, 78] for a more complete list). ML frameworks are interfaces,

	Model	Data preparation - Exploration	Training	Evaluating	Interpretation	Hyperparameters tuning
Classic Supervised	Classification trees	Hierarchical Plots	Tree Network : Hierarchical Plots	Confusion Matrix: Heat Maps ROC plots: line plots Cumulative response curves: line plots Profit curves: line plots	Predictors Importance: Classical bar chart, SHAP summary plot Predictors inter-dependence: classical dependence line plots, Classical 3D dependence plot, SHAP dependence plots SHAP Interaction effects plots	Capture the Hyperparameters and KPIs for each tuning round (KPIs are different for the different models) Combine SPLOM, Line plots and Heatmaps. Slice and dice the data to identify the best combination of parameters for the relevant KPIs.
	Decision Tree Inducers		Confusion Matrix Evolution Plot			
	Bayesian networks					
	Regressions	Dimension Reduction, Clustering Tendency: SPLOM Cluster Tendency: Map Heat Density Based Cluster: Scattered Plot Similarity-Dissimilarity: Scattered Plot	Gradient Decent: Contour Plot (3D and 2D) Confusion Matrix Evalution Plot			
	Support Vector Machines (SVM)		Decision Boundary: Scattered Plot Confusion Matrix Evalution Plot			
Classic Unsupervised	Cluster		Cluster Tendency: Map Heat Density Based Cluster: Scattered Plot Similarity-Dissimilarity: Scattered Plot	Silhouette Coefficient: Clusters silhouette plot Similarity-Dissimilarity: Scattered Plot	Scatter Plots	
	Network Data-Graphs/Link Analysis	Graph plots with color density Motifs aggregation Spectral Partitioning / Scale XY layout	Graph plots with color density Scatter plot matrices	Graphlet frequencies for similarity	Motifs using glyphs Motifs modules Graphlet frequencies for similarity	
Deep Learning	Neural Networks, Genetic, CNN, Recurrent neural network, Deep belief networks	Dimension Reduction, Clustering Tendency: SPLOM Cluster Tendency: Map Heat Density Based Cluster: Scattered Plot Similarity-Dissimilarity: Scattered Plot	Gradient Descent: Contour Plot (3D and 2D) Cluster Tendency: Map Heat Density Based Cluster: Scattered Plot Similarity-Dissimilarity: Scattered Plot Confusion Matrix Evolution Plots Validation loss per step: line plots Validation accuracy per step: line plots Confusion Matrix Evolution Plots	Confusion Matrix: Heat Maps ROC plots: line plots Cumulative response curves: line plots Profit curves: line plots	Scatter Plots Predictors Importance: Classical bar chart, SHAP summary plot Predictors inter-dependence: classical dependence plots, SHAP dependence plots SHAP Interaction effects CARTscan scatter plots Display activation: Image Matrix or Heat Map	
Ensemble Methods	Boosted trees, Bagged trees					

Fig. 15 Taxonomy—visual analytics and the 7 steps of machine learning

libraries, or tools that allow developers to build machine learning models easily and quickly, without needing to code all the underlying algorithms. ML frameworks have collections of pre-built, optimized components. They come with user interfaces that are intended to make it easier to understand, debug, and optimize the ML programs.

Not all the ML frameworks have the same level of interactive features. Some of them only provide a coding platform for the data science practitioner. Others can turn into an end-user application for the expert analyst, including interactive visualization features, such as zooming and filtering different regions of the display, switching between types of visualizations, and moving graphs via drag and drop. This is mainly the case for interactive visualizations with graphs, such as GRAPHVIZ (www.graphviz.org) and D3VIZ from Theano [81]. As the development of interactive visualization frameworks for ML is trying to catch up with the explosion of ML development, many applications and tools are appearing in this domain. However, these visualization tools mainly focus on specific steps of the ML process or on specific models. Examples are BOOSTVis, which is a visual diagnosis tool to help experts analyze and diagnose the training process of tree boosting [40], or iVisClassifier [15] for face recognition.

5 Discussion

One of the big challenges of the adoption of ML in organizations is understanding and interpretation [31]. Trust is hard to get when the ML algorithm is a "black box" [72]. The areas of explainability and interpretability are still emerging. Whenever a new ML model is proposed, questions arise regarding the data used and how the model works and how it will impact the current processes. Visualizations are one of the best ways to interpret and explain the data and the models.

Questions, such as what the data were used to train the model, and why was this data and model combination used, often do not have the straightforward answers one might expect. To partly address this problem, data lineage methods, using visualizations, can be employed to explain how the data were changed and transformed [31].

In many academic fields, algorithms showing high explanatory power are often assumed to be highly predictive [84], but this is not always the case. There are many situations where building the best predictive model differs from building the best explanatory model [84], and modeling decisions often result in trade-offs between the two objectives. When the only objective is to get the best prediction, the goal is quite clear: to find a systemic function that can be treated as a black box, which will result in the lowest average error in the predictions. On the other hand, when the objective is to support "singular, monumental decisions made by businesses, such as how to position a new entrant within a competitive market" [75], the function cannot be completely treated as a black box. In this event, visualizations are and will be a

necessary persuasion tool to convey the message of what and why a specific data-driven decision should be made.

References

1. Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Others, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vi, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng.: TensorFlow: Large-scale machine learning on heterogeneous distributed systems. In: arXiv preprint arXiv:1603.04467, 2016.
2. Muhammad Arif and Saleh Basalamah.: Similarity-dissimilarity plot for high dimensional data of different attribute types in biomedical datasets. In: International Journal of Innovative Computing, Information and Control, 8(2):1275–1297, 2012.
3. Peter Bak and Joachim Meyer.: Effects of cognitive styles and data characteristics on visual data mining. In: Visualization and Data Analysis 2005, volume 5669, pages 77–87. International Society for Optics and Photonics, 2005.
4. Irad Ben-Gal.: Bayesian networks. In: Encyclopedia of Statistics in Quality and Reliability, 1, 2008.
5. H Bischof, A Pinz, and W G Kropatsch.: Visualization methods for neural networks. In: Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol. II. Conference B: Pattern Recognition Methodology and Systems, pages 581–585, Aug 1992.
6. Fernanda Viegas, Martin Wattenberg.: Visualization for Machine Learning. In: Google Brain Team
7. Patrick Breheny and Woodrow Burchett.: Visualization of regression models using visreg. In: The R Journal, 9(2):56–71, 2017.
8. Paul Burkhardt.: Graphing trillions of triangles. In: Information Visualization, 16(3):157–166, 2017.
9. Pablo Casas. : SHAP values for model interpretation. In: GitHub, 2018.
10. Pablo Casas.: A gentle introduction to SHAP values in R. 2019.
11. Tianqi Chen and Carlos Guestrin.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794, 2016.
12. Vladimir Cherkassky and Sauptik Dhar. : Simple Method for Interpretation of High-Dimensional Nonlinear SVM Classification Models. In: DMIN, number January, pages 267–272, 2010.
13. Chih-Hung Chiang, Patrick Shaughnessy, Gary Livingston, and GG Grinstein.: Visualizing graphical probabilistic models. In: UML CS, 2005.
14. F Chollet. : Deep Learning with Python. In: Manning Publications Co., 2018.
15. Jaegul Choo, Hanseung Lee, Jaeyeon Kihm, and Haesun Park.: iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In: 2010 IEEE Symposium on Visual Analytics Science and Technology, pages 27–34. IEEE, 2010.
16. Ana M Cuadros, Fernando Vieira Paulovich, Rosane Minghim, and Guilherme P Telles.: Point Placement by Phylogenetic Trees and its Application to Visual Analysis of Document Collections. In: IEEE VAST, pages 99–106, 2007.
17. Cody Dunne and Ben Shneiderman.: Motif simplification: improving network visualization readability with fan, connector, and clique glyphs. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 3247–3256. ACM, 2013.

18. Holger Ebel, Lutz-Ingo Mielsch, and Stefan Bornholdt.: Scale-free topology of e-mail networks. In: Physical Review E, 66(3):035103, 2002.
19. Qin, Zhuwei and Yu, Fuxun and Liu, Chenchen and Chen, Xiang.: How convolutional neural network see the world-A survey of convolutional neural network visualization methods In: arXiv preprint arXiv:1804.11191, 2018.
20. Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, and Others.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, volume 96, pages 226–231, 1996.
21. Ronak Etemadpour, Robson Motta, Jose Gustavo De Souza Paiva, Rosane Minghim, Maria Cristina Ferreira De Oliveira, and Lars Linsen.: Perception-based evaluation of projection methods for multidimensional data visualization. In: IEEE Transactions on Visualization and Computer Graphics, 21(1):81–94, 2015.
22. Hadi Fanaee-T and Joao Gama.: Event labeling combining ensemble detectors and background knowledge. In: Progress in Artificial Intelligence, 2(2-3):113–127, 2014.
23. Usama Fayyad, G Piatetsky-Shapiro, and Padhraic Smyth.: From data mining to knowledge discovery in databases. In: AI Magazine, pages 37–54, 1996.
24. Foster Provost and Tom Fawcett.: Data Science for Business. In: O'Reilly, 2013.
25. Jerome Friedman, Trevor Hastie, and Robert Tibshirani.: The elements of statistical learning, volume 1. In: Springer Series in Statistics New York, 2001.
26. Thushan Ganegedara.: TensorBoard Tutorial, 2018.
27. Green, Ribarsky, and Fisher.: Visual analytics for complex concepts using a human cognition model. In: 2008 IEEE Symposium on Visual Analytics Science and Technology, pages 91–98, Oct 2008.
28. Yufeng Guo.: The 7 Steps of Machine Learning, 2017. In: towardsdatascience.com
29. Nick Heath.: The top 10 programming languages for machine learning, 2019. In: techrepublic.com
30. Christoph Heindl.: Seamless integration of Matplotlib figures into TensorFlow summaries, 2018.
31. Amy E Hodler, Mark Needham, and Jake Graham. :Artificial Intelligence and Graph Technology Enhancing AI with Context and Connections, 2019. In: neo4j.com
32. Andreas Holzinger. : Human – Computer Interaction and Knowledge Discovery (HCI-KDD): What Is the Benefit of Bringing Those Two Fields to Work Together ? In: A. Cuzzocrea et al. (Eds.): CD-ARES 2013, LNCS 8127 pages 319–328. 2013
33. Alboukadel Kassambara. Practical guide to cluster analysis in R: Unsupervised machine learning, volume 1. STHDA, 2017.
34. D A Keim, F Mansmann, J Schneidewind, and H Ziegler. : Challenges in Visual Data Analysis. In: Tenth International Conference on Information Visualisation IV06, pages:9–16, 2006.
35. Daniel A. Keim. : Information visualization and visual data mining. In: IEEE Transactions on Visualization and Computer Graphics, 7(1):1–8, 2002.
36. Oh-Hyun Hyun Kwon, Tarik Crnovrsanin, and Kwan-Liu Liu Ma.: What would a graph look like in this layout? a machine learning approach to large graph visualization. In: IEEE Transactions on Visualization and Computer Graphics, 24(1):478–488, 2017.
37. Carmen Lai.: User Churn Prediction: A Machine Learning Example, 2016.
38. Chenhui Li, George Baciu, and Yunzhe Wang.: ModulGraph: modularity-based visualization of massive graphs. In: SIGGRAPH Asia 2015 Visualization in High Performance Computing, page 11. ACM, 2015.
39. Tianyi Li, Gregorio Convertino, Wenbo Wang, Haley Most, Tristan Zajonc, and Yi-Hsun Tsai.: HyperTuner: Visual analytics for hyperparameter tuning by professionals. In: Proceedings of the Machine Learning from User Interaction for Visualization and Analytics Workshop at IEEE VIS, 2018.
40. Shixia Liu, Jiannan Xiao, Junlin Liu, Xiting Wang, Jing Wu, and Jun Zhu.: Visual diagnosis of tree boosting methods. In: IEEE Transactions on Visualization and Computer Graphics, 24(1):163–173, 2017.

41. Shusen Liu, Dan Maljovec, Bei Wang, Peer Timo Bremer, and Valerio Pascucci.: Visualizing High-Dimensional Data: Advances in the Past Decade. In: IEEE Transactions on Visualization and Computer Graphics,23(3):1249–1268, 2017.
42. Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee.: Consistent individualized feature attribution for tree ensembles. In: arXiv preprint arXiv:1802.03888, (2), 2018.
43. Scott M Lundberg and Su-in Lee.: A Unified Approach to Interpreting Model Predictions. In: 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA (Section 2):1–10, 2017.
44. Inselberg, Alfred: The plane with parallel coordinates. In: The Visual Computer, Springer, 1(2): 69–91, 1985.
45. Inselberg, Alfred: Visualization and data mining of high-dimensional data. In: Chemometrics and Intelligent Laboratory Systems, Elsevier, 60(1-2): 147–159, 2002.
46. Inselberg, Alfred and Dimsdale, Bernard: Parallel coordinates for visualizing multi-dimensional geometry. In: Computer Graphics, Springer, 1(2): 25–44, 1987.
47. Inselberg, Alfred: Parallel coordinates: visual multidimensional geometry and its applications. In: The Visual Computer, Springer Science & Business Media, 20, 2009.
48. Kandogan, Eser: Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In: Proceedings of the IEEE Information Visualization Symposium, Citeseer, 650, 2000.
49. Wong, Pak Chung and Bergeron, R Daniel: 30 years of multidimensional multivariate visualization. In: Scientific Visualization, 2: 3–33, 1994.
50. Heer, Jeffrey and Bostock, Michael and Ogievetsky, Vadim: A tour through the visualization zoo. In: Communications of the ACM,ACM New York, NY, USA, 6: 59–67, 2010.
51. Heinrich, Julian and Weiskopf, Daniel: State of the Art of Parallel Coordinates. In: Eurographics (STARs), : 95–116, 2013.
52. Müller, Wolfgang and Schumann, Heidrun: Visualization for modeling and simulation: visualization methods for time-dependent data—an overview. In: Proceedings of the 35th Conference on Winter Simulation: Driving Innovation, : 737–745, 2003.
53. Zhang, Yi and Liu, Teng and Li, Kefei and Zhang, Jiawan: Improved visual correlation analysis for multidimensional data. In: Journal of Visual Languages & Computing, Elsevier, 41: 121–132, 2017.
54. Claessen, Jarry HT and Van Wijk, Jarke J: Flexible linked axes for multivariate data visualization. In: IEEE Transactions on Visualization and Computer Graphics, IEEE, 17(12): 2310–231, 2011.
55. Liu, Shixia and Cui, Weiwei and Wu, Yingcai and Liu, Mengchen: A survey on information visualization: recent advances and challenges. In: The Visual Computer, Springer, 30(12): 1373–1393, 2014.
56. Sun, Guo-Dao and Wu, Ying-Cai and Liang, Rong-Hua and Liu, Shi-Xia: A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. In: Journal of Computer Science and Technology, Springer, 28(5): 852–867, 2013.
57. Kovalerchuk, Boris and Grishin, Vladimir: Adjustable general line coordinates for visual knowledge discovery in ND data. In: Information Visualization, SAGE Publications Sage UK: London, England 18(1): 3–32, 2019.
58. Kovalerchuk, Boris: Visual knowledge discovery and machine learning. In: Book - Springer, 2018.
59. Mayo, Matthew.:The 7 Steps of Machine Learning, In: KDnuggets.com, 2018
60. Joachim Meyer, David Shinar, and David Leiser.: Multiple Factors that Determine Performance with Tables and Graphs. In: Human Factors: The Journal of the Human Factors and Ergonomics Society, 39(2):268–286, 1997.
61. Nori, Harsha and Jenkins, Samuel and Koch, Paul and Caruana, Rich.: InterpretML: A unified framework for machine learning interpretability. In arXiv preprint arXiv:1909.09223 (2019).
62. Christoph Molnar.: Interpretable machine learning: A guide for making black box models explainable. In https://christophm.github.io/interpretable-ml-book/(2019).

63. Niels J S Morch, Ulrik Kjems, Lars Kai Hansen, Claus Svarer, Ian Law, Benny Lautrup, Steve Strother, and Kelly Rehm. : Visualization of neural networks using saliency maps. In: Proceedings of ICNN'95-International Conference on Neural Networks, volume 4, pages 2085–2090. IEEE, 1995.
64. Sérgio Moro, Paulo Cortez, and Paulo Rita.: A data-driven approach to predict the success of bank telemarketing. In: Decision Support Systems, 62:22–31, 2014.
65. Tamara Munzner.: In: Visualization Analysis and Design. 2014.
66. Sarang Narkhede.: Understanding AUC - ROC Curve, 2018. In:towardsdatascience.com
67. Chris Nicholson.: Comparison of AI Frameworks. In: SkyMind - A.I. Wiki
68. Joshua O'Madadhain, Danyel Fisher, Padhraic Smyth, Scott White, and Yan-Biao Boey.: Analysis and visualization of network data using JUNG. In: Journal of Statistical Software, 10(2):1–35, 2005.
69. Plotly.: Scatterplot Matrix in Python/SPLOM in R. In: Plotly Graphing Libraries.
70. Sushant : Ratnaparkhi and Milind Paradkar.: Use Decision Trees in Machine Learning to Predict Stock Movements. In: Quant Institute 2017.
71. Donghao Ren, Saleema Amershi, Bongshin Lee, Jina Suh, and Jason D. Williams.: Squares: Supporting interactive performance analysis for multiclass classifiers. In: IEEE Transactions on Visualization and Computer Graphics, 23(1):61–70, 2016.
72. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin.: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. ArXiv Id:1602.049382016.
73. Peter J. Rousseeuw.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. In: Journal of Computational and Applied Mathematics, 20:53–65, Nov 1987.
74. Dominik Sacha, Leishi Zhang, Michael Sedlmair, John A. Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C. North, and Daniel A. Keim. : Visual Interaction with Dimensionality Reduction: A Structured Literature Analysis. In: IEEE Transactions on Visualization and Computer Graphics, 23(1):241–250, 2017.
75. Nathan E Sanders.: A Balanced Perspective on Prediction and Inference for Data Science in Industry. In: Inference in Industrial Data Science 2017.
76. Michael Sedlmair, Andrada Tatu, Tamara Munzner, and Melanie Tory.: A taxonomy of visual cluster separation factors. In: Computer Graphics Forum, volume 31, pages 1335–1344. Wiley Online Library, 2012.
77. Faizan Shaikh. Essentials of Deep Learning: Visualizing Convolutional Neural Networks in Python. In: Analytics Vidhya, 2018.
78. Anton Shaleynikov.: 10 Best Frameworks and Libraries for AI. In: DZone.com / AI Zone, 2018.
79. Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. : Deep inside convolutional networks: Visualising image classification models and saliency maps. In: arXiv preprint arXiv:1312.6034, pages 1–8, 2013.
80. Andrada Tatu, Georgia Albuquerque, Martin Eisemann, Peter Bak, Holger Theisel, Marcus Magnor, and Daniel Keim. : Automated analytical methods to support visual exploration of high-dimensional data. In: IEEE Trans.Vis.Comput.Graph., 17(5):584–597, 2010.
81. Theano.: d3viz – d3viz: Interactive visualization of Theano compute graphs, 2017.
82. Stef Van Den Elzen and Jarke J van Wijk.: BaobabView: Interactive construction and analysis of decision trees. In: 2011 IEEE Conference on Visual Analytics Science and Technology (VAST), pages 151–160. IEEE, 2011.
83. Jake VanderPlas.:Python data science handbook: essential tools for working with data. In: O'Reilly Media, Inc., 2016.
84. Phoebe Wong. Predicting vs. Explaining And Why Data Science Needs More "Half-Bayesians", In: Towards DataScience 2019.

Explainable Artificial Intelligence (XAI): Motivation, Terminology, and Taxonomy

Aviv Notovich, Hila Chalutz-Ben Gal, and Irad Ben-Gal

1 Motivation

Deep learning algorithms and deep neural networks (DNNs) have become extremely popular due to their high-performance accuracy in complex fields, such as image and text classification, speech understanding, document segmentation, credit scoring, and facial recognition. As a result of the highly nonlinear structure of deep learning algorithms, these networks are hard to interpret; thus, it is not clear how the models reach their conclusions and therefore, they are often considered black-box models. The poor transparency of these models is a major drawback despite their effectiveness. In addition, recent regulations such as the General Data Protection Regulation (GDPR), require that, in many cases, an explanation will be provided whenever the learning model may affect a person's life. For example, in autonomous vehicle applications, methods for visualizing, explaining, and interpreting deep learning models that analyze driver behavior and the road environment have become standard. Explainable artificial intelligence (XAI) or interpretable machine learning (IML) programs aim to enable a suite of methods and techniques that produce more explainable models while maintaining a high level of output accuracy [1–4]. These programs enable human users to better understand, trust, and manage the emerging generation of artificially intelligent systems [4].

Many people do not feel comfortable when blindly agreeing with an AI system's decisions in various situations, without some understanding of the decision-making process used by such a system. To achieve trust in AI systems, detailed "explana-

A. Notovich · I. Ben-Gal (✉)
Department of Industrial Engineering, Tel Aviv University, Tel-Aviv, Israel
e-mail: bengal@tauex.tau.ac.il

H. Chalutz-Ben Gal
School of Industrial Engineering and Management, Afeka Tel Aviv Academic College of Engineering, Tel Aviv, Israel

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_41

tions" of AI system decisions seem necessary. Such explanations provide insights into and interpretability of the rationale of the applied AI algorithms and help users trust the system conclusions. As ML and AI modeling are increasingly involved in critical areas such as transportation, retail, insurance, medicine, criminal justice, and financial markets, it seems vital that these models become more easily understood [9].

The XAI-related concepts of explainability, interpretability, and accuracy are presented next, followed by segmentation of XAI methods.

2 Explainability, Interpretability, and Related XAI Terms

The definitions of both AI explainability and AI interpretability have multiple meanings and sometimes there is little to no consensus in the research community regarding these terms [1]. There are a few conflicting definitions that differ from each other in terms of theme and community. In particular, various AI-related communities approach the concept of explainability from different angles. The term explainable AI (XAI) has a double meaning itself. Sometimes it is used to represent methods that help explore the mechanisms of the AI methods or the AI systems themselves; for example, a researcher may seek an interpretation of how these methods or systems work or which features are important when making predictions. In other cases, the term XAI is related to explanations about particular inputs, outputs and examples, such as understanding how a record in a dataset was mapped to a specific segment or recommendation.

Lipton [9] addresses this ambiguity and claims that many XAI papers provide diverse and sometimes non-overlapping motivations for interpretability and offer myriad notions of what makes render models interpretable. Despite such ambiguity, many papers proclaim interpretability axiomatically, absent further explanation.

Explainability and interpretability are closely related concepts in the literature. Sometimes, the term "explainability" refers to "why" a recommendation has been made, while the term "interpretability" refers to "how" that recommendation was obtained [2]. Accordingly, it has been claimed that interpretability is one of the approaches that achieves explainability [3]. Explainable AI (XAI) aims to develop tools that are able to explain AI model decisions to inexpert users. To do so, the model might be either interpretable or non-interpretable. Interpretable models try to develop models whose decision mechanism is locally or globally transparent. Therefore, the model outputs are usually naturally explainable.

Other approaches claim that "Explainability" and "Interpretability" are two related, yet distinct, concepts when referring to AI systems.

"AI Explainability" refers to the ability of an AI or ML model to provide understandable and clear explanations for its predictions or decisions. An explainable model should be able to articulate the reasons behind the model outputs in an easy and comprehended way by human users. For example, explainability is particularly important in domains where the impact of AI decisions can have legal, ethical,

or societal consequences (e.g., healthcare, finance, and autonomous vehicles). An explainable model contributes to trust the model's decision-making process.

"AI Interpretability," on the other hand, refers to the ability of a model to be understood by humans in terms of its internal robustness or how it arrives at its outputs. An interpretable model is one that can be explained in terms of its feature importance, decision rules, or other transparent representations, which allow humans to understand how the model arrives at its predictions. Interpretability is often used interchangeably with explainability, but it can also refer specifically to the technical characteristics of a model that make it transparent and understandable.

Even though interpretability and explainability have been used interchangeably, Došilović et al. [4] claim it is important to distinguish between them. As such, explainable models are interpretable by default, but the reverse is not always true. However, interpretability is not the only way to achieve explainability. There are models that reveal their internal decision mechanisms for explanation purposes and use complex explanation techniques, such as neural attention mechanisms [3].

According to Došilović et al. [4], interpretability alone is insufficient. To increase human trust in black-box methods, it is necessary to develop explainability models that summarize the reasons for the model output. The authors assume that, while both mechanisms are important, interpretability is a substantial first step that provides the capacity to defend model actions and recommendations, provide relevant responses to questions, and be audited.

Interpretable models encompass much of the present work in explainable AI [1]. The main reason is the increased usage of deep neural networks that are so hard to interpret. However, it is still challenging to formulate a line of reasoning that explains a model's decision-making process to the user while relying on human-understandable features of the input data. Nonetheless, reasoning is a critical step when formulating an explanation about why or how an AI-based recommendation has been made.

To summarize the above discussion, despite the inherent inconsistency that one can find in the literature, the following list presents some of the common terms and their popular explanation in the XAI community. The list is mainly based on [2–4] that provide an excellent overview on the topic.

- **Interpretability –** users should be able to understand and reason about the model output.
- **Model Transparency –** defined in terms of *simulatability, decomposability,* and *algorithmic transparency.*
- **Simulatability –** whether a human can use the input data together with the model to reproduce every calculation necessary to make the prediction.
- **Decomposability –** whether there is an intuitive explanation of all the model parameters.
- **Algorithmic Transparency –** an ability to explain how the learning algorithm works.
- **Model Functionality –** defined in terms of *textual description, visualization,* and *local explanation.*

- **Textual Description –** a semantically meaningful description of the model output.
- **Visualization –** a method for explaining a model through visualization of its output and its parameters.
- **Local Explanation –** rather than explaining the mapping of an entire model, local changes are introduced using specific input vectors for a given output class. Explanation is provided on specific use cases or instances.
- **Global Interpretability –** understanding the entire ML model behavior, holistic reasoning that leads to all different possible outcomes.
- **Local Interpretability –** understanding a single model prediction.
- **Activation Maximization –** generation of an input image that maximizes the filter output activations.
- **Anchor –** rule that sufficiently "anchors" the prediction locally such that changes to the rest of the instance's feature values do not matter.
- **Surrogate Model –** a simple model on top of (or besides) a complex model, trained based on the same input and the same predictions of the original complex model in order to mimic a better explanation and interpretation.
- **Partial Dependence Plot (PDP) –** a graphical representation that helps visualize the average partial relationship between one or more input variables and the predictions of a complex model.
- **Individual Conditional Expectation (ICE) –** a graphical representation that reveals interactions and individual differences by separating the PDP output.
- **Knowledge Extraction –** the task of extracting explanations/knowledge from the complex model during training and encoding that knowledge as an internal representation of a complex model.
- **Influence Methods –** several techniques that carefully modify the inputs and measure how much the prediction changed according to each modification.
- **Example-Based Explanation –** selection of specific data points to explain the behavior of machine learning models.

3 Accuracy and Explainability

A conventional statement is that there is an inherent trade-off between model explainability and model effectiveness, thus stating that one can either achieve high explainability with simpler models or high accuracy with more complex models, which are generally harder to interpret [3]. Figure 1 presents a possible schematic view of the trade-off between the model explainability and the model effectiveness. Similar graphs can be found in many papers, with the same message.

This belief raises a common dilemma among practitioners regarding whether to choose an understandable/explainable simple algorithm, while sacrificing prediction accuracy or to choose an accurate latent factorization modeling approach, while sacrificing explainability [5]. However, there is also a belief that these two goals

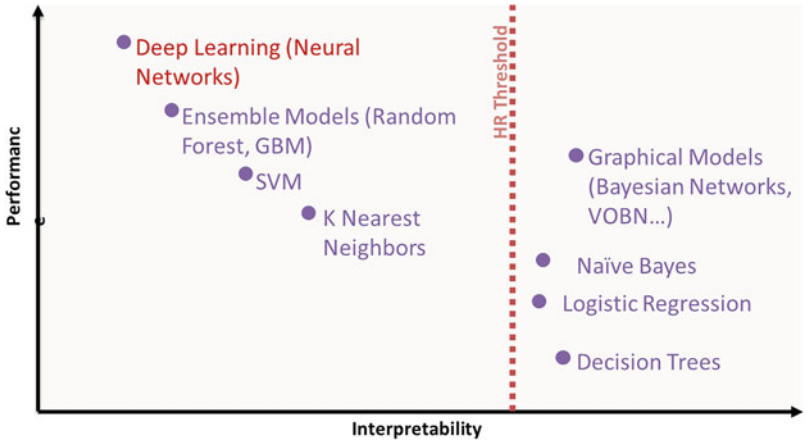


Fig. 1 The inherent trade-off between model performance to model interpretability

do not necessarily contradict each other [6], which claims that this assumption is primarily relevant for cases related to structured data with meaningful features.

Extensive research has focused on state-of-the-art techniques, such as deep learning approaches, which emphasize a model design that is both effective and explainable. Developing explainable deep models is thus an attractive direction in the broader AI community, leading to progress in essential explainable machine learning problems [3].

4 Segmentation of XAI Approaches

There are several ways to classify and segment the different XAI approaches [1, 2, 4]. Adadi and Berrada [3] propose a categorization for XAI methods that considers the model's complexity of interpretability, scope of interpretability, and level of dependency. In the next sections, we follow earlier surveys by Chakraborty [2], as well as Adadi and Berrada [3].

4.1 Complexity-Related Methods

Many works in the literature assume that model complexity is directly related to interpretability. Thus, simpler models are easier to interpret. Accordingly, to better interpret complex models, there is a need to introduce a simpler surrogate model or an algorithm for interpretability. Several works following this direction are described in this section.

Xu et al. [8] consider the task of automatically generating image captions as a goal that is central to scene understanding. The authors introduce an attention-based image caption model that automatically learns how to describe image content. They train the attention model using standard backpropagation techniques over deep neural networks and by maximizing a variational lower bound. The proposed model gains insight and interpretation by visualizing "where" and "what" the attention is focused on. Relying on visualization and benchmark datasets, they demonstrate how their model is able to interpret the images.

Caruana et al. [7] deal with pneumonia risk prediction by applying generalized additive models with pairwise interactions (GA^2Ms). The proposed model achieves state-of-the-art accuracy and is able to uncover surprising patterns in the data that previously challenged researchers and prevented the implementation of complex machine learning models in this domain. The model is used to identify and remove such patterns to obtain a better performance.

Letham et al. [6] propose a method based on decision trees called Bayesian Rule Lists (BRL), which produces a predictive model that is not only accurate but also interpretable to human experts. This model generates conditional "if/then" statements (e.g., "if high blood pressure, then stroke") that discretize a high-dimensional, multivariate feature space into a series of simple, readily interpretable decision statements. Such an outcome is highly interpretable and provides concise and convincing capabilities that are able to gain the trust of domain experts.

Lipton [9] proposes following a post hoc explanation approach with two stages. This approach first allows complex, uninterpretable black-box models to generate high-performance outputs, and then it applies a separate set of techniques to obtain explainability and interoperability over the outputs. Such an approach views the interpretability task as a reverse engineering process that provides the required explanations without altering or even knowing the inner works of the original black-box model.

4.2 Global and Local Interpretability Approaches

There are two primary approaches when seeking explainability and interpretability for AI and ML models. The first is the global interpretability approach, which aims to provide a systematic view and general understanding of the AI system in use. Thus, the global interpretability approach seeks a complete view of the decisions and operations of the entire AI model. For example, this approach focuses on explaining the overall model analysis using a set of rules and measures that determine the global feature importance and explain the model outcomes. Such explainability could be used for example by technical experts to obtain a better modeling decision.

The second is the local interpretability approach, which is focused on approximating and explaining individual predictions and case-by-case outcomes. Thus, unlike global interpretability, it does not seek to explain the whole model but rather the specific outcomes of the AI system under different feature values and conditions.

For example, local interpretability can be used to explain and justify an AI system recommendation that a specific client is not entitled to a bank loan due to his income level and previous loans or other personal financial conditions. As such, global interpretability is often better for non-technical users.

Note that some studies aim to combine global and local interpretability; examples of this approach include Guidotti et al. [21] and Linsley et al. [22].

Global Interpretability

As indicated, the goal of the global XAI approach is to understand the entire logic of a model and the entire pattern of reasoning that leads to different possible outcomes. This approach is most relevant in situations that require a high level of accountability and justification, such as AI applications in medical domains [1]. In these cases, a global effect estimate is often more helpful than many separate explanations for different possible predictions. Some examples that imply such an approach are as follows.

Nguyen et al. [12] aim to study what each of a DNN's neurons is learning to detect. They use activation maximization (AM), which synthesizes an input (e.g., an image) that highly activates a neuron. The proposed method generates synthetic images and reveals the features learned by each neuron in an interpretable way.

Valenzuela-Escárcega et al. [11] propose a supervised approach for information extraction, which combines bootstrapping with representation learning. The proposed algorithm iteratively learns custom embeddings for multi-word entities and their matched patterns from example entities for each classification category. This approach outputs a globally interpretable model consisting of a decision list that acts as an interpretation of the model.

Yang et al. [10] propose a method that interprets black-box machine learning models globally using a binary interpretation tree. The interpretation tree explicitly represents the most important decision rules that are implicitly contained in the black-box machine learning models. The proposed learning algorithm partitions the input variable space by maximizing the difference between the average contributions of the split variable over the divided spaces. This method results in a contribution matrix that consists of the contributions of input variables to the predicted scores for each single prediction. The authors demonstrate the effectiveness of their method for diagnosing machine learning models over multiple tasks as well as for analyzing the models in terms of human understanding.

Local Interpretability

The goal of local interpretability is to explain the reasons for a specific decision or single prediction that the ML model has made. Here, we discuss research works focused on this type of interpretability.

Ribeiro et al. [13] proposed a novel technique to explain the predictions of a classifier in an interpretable and faithful manner, by *locally* learning an interpretable model based on individual predictions. They called it the *local interpretable model-agnostic explanation* (LIME). The method, which is formulated as a submodular

optimization problem, approximates a black-box model locally in the neighborhood of any prediction.

Ribeiro et al. [14] extend LIME using decision rules called "anchors". An anchor explanation is a rule that sufficiently "anchors" the prediction locally, such that changes to the rest of an instance's feature values do not affect the AI system recommendation.

A similar approach was used in a series of studies [15–19] that analyzed image classification by a family of ML models. In particular, the analyses identified image regions (pixels) that were found to be particularly influential on the final classification. Several names were given to this approach, including *sensitivity maps*, *saliency maps*, or *pixel attribution maps*. These techniques assign an "importance" score to individual pixels, which is meant to reflect their influence on the final classification of the image. A similar yet opposing concept is applied in *adversarial learning*, which aims to find and modify these specific pixels as a means to distort and change the correct classification [40].

Lundberg and Lee [20] present a unified framework for interpreting predictions, named SHAP (SHapley Additive exPlanations). SHAP assigns each particular prediction's features an importance value. Its novel components include (i) the identification of a new class of additive feature importance measures and (ii) theoretical results showing that there is a unique solution in this class with a set of desirable properties. Additionally, the authors show that by using different kernels, SHAP can be model agnostic.

According to surveys by Chakraborty et al. [2] and by Adadi and Berrada [3], local explanations are the most commonly used explanation methods in XAI and are particularly applied to DNN models.

4.3 Model-Related Methods

Another popular way to classify model interpretability techniques is according to whether they are model agnostic or model-specific; model-agnostic methods can be applied to any model type, while model-specific methods work only for specific models.

Model-Specific Interpretability
As model-specific techniques are limited to a particular model; according to [2, 3], they are less popular than model-agnostic interpretability methods, which often generate more interest.

Model-Agnostic Interpretability
According to Mary [4], a specific class of model-agnostic methods is related to those that can be applied primarily to black box models' inputs and outputs. The usability and popularity of these methods can be found by examining a variety of use cases [5]. This class of methods addresses prediction tasks and explanation tasks separately. Model-agnostic interpretations are usually post hoc, i.e., they are

generally applied to interpret DNNs and could be either local or global interpretable models [3]. Herein, we present an overview of the studies focused on model-agnostic interpretability, grouped by the applied techniques. In particular, one can find four primary technique types: *visualization, knowledge extraction*, *influence methods,* and *example-based explanation* [3].

4.3.1 Visualization

One way to illustrate and better understand an ML model output, especially a DNN, is to represent it visually; for example, researchers have previously explored hidden patterns within a segment of the neural network (including a single neural unit). Many visualization techniques are applied to supervised learning models in which the active neurons and pixels can be highlighted per labeled class. The literature contains three primary types of explainability techniques that are related to visualization: *Surrogate models*, *Partial Dependence Plots* (PDP), and *Individual Conditional Expectation* (ICE) [3].

Surrogate Models

Surrogate modeling refers to building a simple model (e.g., a linear model or decision tree) to approximate a more complex model (e.g., a DNN) to help explain how the complex model reaches its decisions. To build a surrogate model, one should often train the simpler model based on the inputs and the outputs of the more complex original model. In many cases, the simpler model's output can be visualized to further highlight the important features on the model output. This technique is sometimes useful; however, there is no theoretical guarantee that this technique will produce a clean and effective explanation for the complex model.

LIME [13] is a popular method for constructing local surrogate models around subsets of observations. Bastani et al. [23] built such a surrogate model approach by extracting a decision tree that represents a complex model's behavior. Thiagarajan et al. [24] proposed an approach for building the "TreeView" representation using a surrogate model that performs hierarchical partitioning of the feature space. This surrogate model reveals the iterative rejection of unlikely class labels until the correct association is predicted.

Partial Dependence Plot (PDP)

The partial dependence plot is another graphical representation that helps visualize the average partial relationship between one or more input variables and a complex model's predictions. PDP has been used in several studies to understand the relationship between predictors and inputs under several conditions (e.g., [25–27]).

Individual Conditional Expectation (ICE)

Individual conditional expectation (ICE) can be considered an extension of PDP. ICE plots reveal interactions and individual differences by separating the PDP output. ICE has been used in several studies (e.g., [28, 29]), in which the advantage of ICE over PDP has been demonstrated and analyzed.

4.3.2 Knowledge Extraction

Knowledge extraction (KE) refers to the task of extracting explanations and knowledge from a complex model during the training phase and encoding it as an internal representation of a complex model. In the literature, two primary KE techniques include *rule extraction and model distillation* [3].

Rule Extraction

Rule extraction (RE) aims to find rules that provide approximation of the decision-making process for a more complex model. In a sense, it is similar to the association rules that were used in data-mining tasks to extract simple rules from ML classification models.

Using RE, one can obtain a better description of the knowledge learned by the complex model during training. Several studies have implemented rule extraction (e.g., [30, 31]).

Model Distillation

Model distillation (MD) is based on model compression techniques. MD was originally proposed to reduce the computational cost of a model at runtime but was later targeted at interpretability. Distillation is a model compression that transfers information from deep networks to shallow networks in the form of "teacher to student" [32]. Several studies have implemented model distillation (e.g., [33, 34]).

4.3.3 Influence Methods

Influence methods refer to several techniques that systematically modify a model's inputs and then measure how much the prediction changed according to each modification. In this way, a relevance score for each feature is computed. According to [3], the literature describes three alternative methods for obtaining the input variable's relevance: *sensitivity analysis*, *layer-wise relevance propagation,* and *feature importance*. These approaches are discussed next.

Sensitivity Analysis

Zhang and Wallace [35] introduce a sensitivity measure that determines how a complex model's output is influenced by its input and/or weight perturbations. In particular, they conducted a sensitivity analysis examining one-layer convolutional neural networks (CNNs) to explore the effect of architecture components on model performance; their aim was to distinguish between important and comparatively inconsequential design decisions for sentence classification.

Sensitivity analysis (SA) is widely used to verify whether model outputs remain stable when the data are changing and to support robustness verification in general. Cortez and Embrechts [36] proposed a global SA (GSA) method, which extends the applicability of previous SA methods and several visualization techniques when assessing input relevance and effects on the model's responses. The authors demonstrate the GSA method's capabilities by conducting several experiments using an NN ensemble and SVM model and including both synthetic and real-world datasets. It is worth mentioning, however, that this approach produces an explanation only over the variation of the function values but not the function itself.

Layer-Wise Relevance Propagation (LRP)

Bach et al. [37] proposed a pixel-wise decomposition for nonlinear classifiers. This technique provides visualization of the contributions of single pixels to the predictions of kernel-based classifiers, which can be visualized using heat maps. The proposed technique focuses the analysis on regions of potential interest while tracing backward from the prediction to the input layer. Unlike sensitivity analysis, this technique explains the predictions relative to the state of maximum uncertainty.

Feature Importance

Feature importance provides a score for each input feature that represents its contribution to the predictions of a complex ML model. Basically, this technique generates a permutation of the input features and measures the corresponding model error. Features with high importance increase the model error more significantly when permutated than a feature with low importance. Fisher et al. [38] proposed a technique called model class reliance (MCR), which sets the range of feature importance values across several models for a pre-specified class. Casalicchio et al. [29] used SHAP values to generate a feature importance score for every input feature.

4.3.4 Example-Based Explanation

Example-based explanations (EBEs) are techniques that select particular data points from the dataset to explain the behavior of the ML/AI model. In the reviewed literature, the two primary EBE techniques are *prototypes and criticisms* and *counterfactual explanations* [3].

Prototypes and Criticisms

To avoid overfitting the learning model, a strong representation for the data points must be selected. Kim et al. [39] claim that although example-based explanations are often used to interpret highly complex distributions, prototypes alone rarely sufficiently represent the essence of the model complexity. Motivated by the Bayesian model criticism framework, they develop the MMD-critic, which efficiently learns prototypes and criticism designed to aid human interpretability. The authors evaluate the prototypes selected by MMD-critic using a nearest prototype classifier, demonstrating competitive performance when compared to baselines.

Counterfactual Explanations

Counterfactual explanations attempt to find the boundary at which the learning model will change its decision or recommendation with minimum conditions. This outcome is achieved without the need to describe the algorithm's full logic. Yuan et al. [40] noted that ML models are vulnerable to well-designed input samples, called adversarial examples. Adversarial examples may be invisible to humans but can fool a complex ML model and alter their decision with minimal change to the input. The authors review recent findings related to adversarial examples for deep neural networks, summarize the methods that generate adversarial examples, and propose a taxonomy for these methods.

5 Final Remark

Adadi and Berrada [3] provide an excellent summary of different XAI methods. In a summary taxonomy table, they classify various XAI models and techniques by analyzing whether they are intrinsic/post hoc, global/local, and model specific/model agnostic.

As claimed by the authors, XAI is a vital interdisciplinary research direction and a major building block in the AI ecosystem. The potential impact of XAI can affect various new applications in areas such as transportation, healthcare, military, retail, legal, finance, and well-being. Yet, despite its importance, XAI research is still unstructured, and the human aspects in it can be further studied.

References

1. Erico Tjoa, Cuntai Guan, "A Survey on Explainable Artificial Intelligence (XAI): towards Medical XAI" 2019. [Online] https://arxiv.org/pdf/1907.07374.pdf
2. Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, Moustafa Alzantot, Federico Cerutti, Mani Srivastava, Alun Preece, Simon Julier, Raghuveer M. Rao, Troy D. Kelley, Dave Braines, Murat Sensoyk, Christopher J. Willis, Prudhvi Gurram IBM T. J. Watson Research Center, Crime and Security Research Institute, Cardiff University, UCLA, IBM UK, Army Research Lab, Adelphi, Ozyegin University, BAE Systems AI Labs University College London "Interpretability of Deep Learning Models: A Survey of Results" 2017. [Online] https://orca.cf.ac.uk/101500/1/Interpretability%20of%20Deep%20Learning%20Models%20-%20A%20Survey%20of%20Results.pdf
3. Amina Adadi, Mohammed Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)" 2018. [Online] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8466590
4. Filip Karlo Došilović, Mario Brcic, Nikica Hlupic "Explainable Artificial Intelligence: A Survey" 2018. [Online] https://www.researchgate.net/publication/325398586_Explainable_Artificial_Intelligence_A_Survey
5. Jakob M. Schoenborn, Klaus-Dieter Althof, "Recent Trends in XAI: A Broad Overview on current Approaches, Methodologies and Interactions". 2019. [Online] http://gaia.fdi.ucm.es/events/xcbr/papers/XCBR-19_paper_1.pdf
6. B. Letham, C. Rudin, T. H. McCormick, and D. Madigan, "Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model," Ann. Appl. Statist., vol. 9, no. 3, pp. 1350–1371, 2015. [Online] https://arxiv.org/pdf/1511.01644.pdf
7. R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2015, pp. 1721–1730. [Online] http://people.dbmi.columbia.edu/noemie/papers/15kdd.pdf
8. K. Xu et al., "Show, attend and tell: Neural image caption generation with visual attention," in Proc. Int. Conf. Mach. Learn. (ICML), 2015, pp. 1–10. [Online] https://arxiv.org/pdf/1502.03044.pdf
9. Z. C. Lipton, "The mythos of model interpretability," in Proc. ICML Workshop Hum. Interpretability Mach. Learn., 2016, pp. 96–100. [Online] https://arxiv.org/pdf/1606.03490.pdf
10. C. Yang, A. Rangarajan, and S. Ranka. (2018). "Global model interpretation via recursive partitioning." [Online] https://arxiv.org/pdf/1802.04253.pdf
11. M. A. Valenzuela-Escárcega, A. Nagesh, and M. Surdeanu. (2018). "Lightly-supervised representation learning with global interpretability." [Online] https://arxiv.org/pdf/1805.11545.pdf
12. A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2016, pp. 3387–3395. [Online] https://arxiv.org/pdf/1605.09304.pdf
13. M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should i trust you?': Explaining the predictions of any classifier," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 1135–1144. [Online] https://www.kdd.org/kdd2016/papers/files/rfp0573-ribeiroA.pdf
14. M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in Proc. AAAI Conf. Artif. Intell., 2018, pp. 1–9. [Online] https://homes.cs.washington.edu/~marcotcr/aaai18.pdf
15. K. Simonyan, A. Vedaldi, and A. Zisserman. (2013). "Deep inside convolutional networks: Visualising image classification models and saliency maps. [Online] https://arxiv.org/pdf/1312.6034.pdf

16. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in Proc. Eur. Conf. Comput. Vis. Zurich, Switzerland: Springer, 2014, pp. 818–833. [Online] https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf
17. B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and O. Torralba, "Learning deep features for discriminative localization," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., June 2016, pp. 2921–2929. [Online] https://arxiv.org/abs/1512.04150
18. M. Sundararajan, A. Taly, and Q. Yan. (2017). "Axiomatic attribution for deep networks." [Online] https://arxiv.org/pdf/1703.01365.pdf
19. D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. (2017). "SmoothGrad: Removing noise by adding noise." [Online] https://arxiv.org/pdf/1706.03825.pdf
20. S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 4768–4777. [Online] https://arxiv.org/pdf/1705.07874.pdf
21. R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. (2018). "Local rule-based explanations of black box decision systems." [Online] https://arxiv.org/pdf/1805.10820.pdf
22. D. Linsley, D. Scheibler, S. Eberhardt, and T. Serre. (2018). "Globaland-local attention networks for visual recognition." [Online] https://arxiv.org/pdf/1805.08819.pdf
23. O. Bastani, C. Kim, and H. Bastani. (2017). "Interpretability via model extraction. [Online] https://arxiv.org/pdf/1706.09773.pdf
24. J. J. Thiagarajan, B. Kailkhura, P. Sattigeri, and K. N. Ramamurthy. (2016). "TreeView: Peeking into deep neural networks via feature-space partitioning." [Online] https://arxiv.org/pdf/1611.07429.pdf
25. D. P. Green and H. L. Kern, "Modeling heterogeneous treatment effects in large-scale experiments using Bayesian additive regression trees," in Proc. Annu. Summer Meeting Soc. Political Methodol., 2010, pp. 1–40. [Online] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.190.3826&rep=rep1&type=pdf
26. J. Elith, J. Leathwick, and T. Hastie, "A working guide to boosted regression trees," J. Animal Ecol., vol. 77, no. 4, pp. 802–813, 2008. [Online] https://besjournals.onlinelibrary.wiley.com/doi/full/10.1111/j.1365-2656.2008.01390.x
27. S. H. Welling, H. H. F. Refsgaard, P. B. Brockhoff, and L. H. Clemmensen. (2016). "Forest floor visualizations of random". [Online] https://arxiv.org/pdf/1605.09196.pdf
28. A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation," J. Comput. Graph. Statist., vol. 24, no. 1, pp. 44–65, 2015, [Online] https://www.tandfonline.com/doi/abs/10.1080/10618600.2014.907095
29. G. Casalicchio, C. Molnar, and B. Bischl. (2018). "Visualizing the feature importance for black box models." [Online] https://arxiv.org/pdf/1804.06620.pdf
30. U. Johansson, R. König, and I. Niklasson, "The truth is in there—Rule extraction from opaque models using genetic programming," in Proc. FLAIRS Conf., 2004, pp. 658–663. [Online] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.470.4124&rep=rep1&type=pdf
31. T. Hailesilassie. (2017). "Rule extraction algorithm for deep neural networks: A review." [Online] https://arxiv.org/abs/1610.05267
32. P. Sadowski, J. Collado, D. Whiteson, and P. Baldi, "Deep learning, dark knowledge, and dark matter," in Proc. NIPS Workshop High-Energy Phys. Mach. Learn. (PMLR), vol. 42, 2015, pp. 81–87. [Online] http://proceedings.mlr.press/v42/sado14.pdf
33. S. Tan, R. Caruana, G. Hooker, and Y. Lou. (2018). "Detecting bias in black-box models using transparent model distillation." [Online] https://arxiv.org/abs/1710.06169
34. Z. Che, S. Purushotham, R. Khemani, and Y. Liu. (2015). "Distilling knowledge from deep networks with applications to healthcare domain." [Online] https://arxiv.org/abs/1512.03542
35. Y. Zhang and B. Wallace (2016). "A sensitivity analysis of (and practitioners' Guide to) convolutional neural networks for sentence classification. [Online] https://arxiv.org/abs/1510.03820

36. P. Cortez and M. J. Embrechts, "Using sensitivity analysis and visualization techniques to open black box data mining models," Inf. Sci., vol. 225, pp. 1–17, Mar. 2013. [Online] https://core.ac.uk/download/pdf/55616214.pdf
37. S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," PLoS ONE, vol. 10, no. 7, p. e0130140, 2015. [Online] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0130140
38. A. Fisher, C. Rudin, and F. Dominici. (2018). "Model class reliance: Variable importance measures for any machine learning model class, from the 'rashomon' perspective." [Online] https://arxiv.org/abs/1801.01489
39. B. Kim, R. Khanna, and O. O. Koyejo, "Examples are not enough, learn to criticize! criticism for interpretability," in Proc. 29th Conf. Neural Inf. Process. Syst. (NIPS), 2016, pp. 2280–2288 [Online] https://papers.nips.cc/paper/6300-examples-are-not-enough-learn-to-criticize-criticism-for-interpretability.pdf
40. X. Yuan, P. He, Q. Zhu, and X. Li. (2017). "Adversarial examples: Attacks and defenses for deep learning." [Online] https://arxiv.org/abs/1712.07107