

Annalisa Appice
Corrado Loglisci
Giuseppe Manco
Elio Masciari
Zbigniew W. Ras (Eds.)

LNAI 10785

New Frontiers in Mining Complex Patterns

6th International Workshop, NFMCP 2017
Held in Conjunction with ECML-PKDD 2017
Skopje, Macedonia, September 18–22, 2017
Revised Selected Papers

Lecture Notes in Artificial Intelligence

10785

Subseries of Lecture Notes in Computer Science

LNAI Series Editors

Randy Goebel

University of Alberta, Edmonton, Canada

Yuzuru Tanaka

Hokkaido University, Sapporo, Japan

Wolfgang Wahlster

DFKI and Saarland University, Saarbrücken, Germany

LNAI Founding Series Editor

Joerg Siekmann

DFKI and Saarland University, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/1244>

Annalisa Appice · Corrado Loglisci
Giuseppe Manco · Elio Masciari
Zbigniew W. Ras (Eds.)

New Frontiers in Mining Complex Patterns

6th International Workshop, NFMCP 2017
Held in Conjunction with ECML-PKDD 2017
Skopje, Macedonia, September 18–22, 2017
Revised Selected Papers

Editors

Annalisa Appice
University of Bari Aldo Moro
Bari
Italy

Corrado Loglisci
University of Bari Aldo Moro
Bari
Italy

Giuseppe Manco
CNR
Rende
Italy

Elio Masciari
CNR
Rende
Italy

Zbigniew W. Ras
University of North Carolina
Charlotte, NC
USA

and
Polish Japanese Academy of Information
Technology
Warsaw
Poland

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Artificial Intelligence
ISBN 978-3-319-78679-7 ISBN 978-3-319-78680-3 (eBook)
<https://doi.org/10.1007/978-3-319-78680-3>

Library of Congress Control Number: 2018937389

LNCS Sublibrary: SL7 – Artificial Intelligence

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

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

Preface

Modern automatic systems are able to collect huge volumes of data, often with a complex structure (e.g., multi-table data, network data, Web data, time series and sequences, trees and hierarchies). Massive and complex data pose new challenges for current research in data mining. Specifically, they require new models and methods for their storage, management, and analysis, in order to deal with the following complexity factors:

- Data with a complex structure (e.g., multi-relational, time series and sequences, networks, and trees) as input or output of the data mining process
- Data collections with many examples and/or many dimensions, where data may be processed in (near) real time
- Partially labeled data
- Data which arrive continuously as a stream, at high rate, subject to concept drift

The 6th International Workshop on New Frontiers in Mining Complex Patterns (NFMCP 2017) was held in Skopje (Macedonia) in conjunction with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2017) on September 22, 2017. The purpose of this workshop was to bring together researchers and practitioners of data mining who are interested in the latest developments in the analysis of complex and massive data sources, such as blogs, event or log data, medical data, spatiotemporal data, social networks, mobility data, sensor data, and streams. The workshop was aimed at discussing and introducing new algorithmic foundations and representation formalisms in complex pattern discovery. Finally, it encouraged the integration of recent results from existing fields, such as statistics, machine learning, and big data analytics. This book features a collection of revised and significantly extended versions of papers accepted for presentation at the workshop. These papers went through a rigorous review process to ensure compliance with Springer's high-quality publication standards. The individual contributions of this book illustrate advanced data mining techniques which take advantage of the informative richness of both complex data and massive data for efficient and effective identification of complex information units present in such data.

The book is composed of 13 chapters.

Chapter 1 introduces an efficient algorithm to analyze pharmacogenomic data and discover association rules between gene variants of a patient and drug-dependent adverse events.

Chapter 2 describes a classification-based approach for speech remediation. This is used to identify which portions of a speech can be deleted, in order to enhance the speech understandability in terms of both speech content and speech flow.

Chapter 3 presents a heterogeneous clustering algorithm that is able to predict possibly unknown lncRNA–disease relationships by analyzing complex heterogeneous biological networks.

Chapter 4 illustrates a probabilistic generative model, in order to address the problem of positive-unlabeled learning by considering a set of positive samples and a (usually larger) set of unlabeled ones.

Chapter 5 proposes a constraint programming approach that is combined with large neighborhood search, in order to efficiently identify homogeneous subsets of genes, which are similarly expressed across subsets of patients.

Chapter 6 investigates the use of the scaled correlation function, in order to derive the structure of a functional brain network from the activity series associated with the network nodes.

Chapter 7 considers the problem of manufacturing defect identification. It compares two approaches that address the multiple instance problem when the traditional instance localization assumption is not met.

Chapter 8 focuses on the problem of designing ensemble strategies for defending the company's brands from an unauthorized use.

Chapter 9 investigates the task of electricity load forecasting through unsupervised ensemble learning of clustered time series data.

Chapter 10 tackles the problem of phenotype traits prediction using supervised and semi-supervised classification trees as well as supervised and semi-supervised random forests of classification trees.

Chapter 11 introduces an approach that constructs a label hierarchy as a decomposition of the output space of a classification problem, in order to improve the predictive performance.

Chapter 12 illustrates a non-parametric Bayesian approach, in order to fit a mixture model of Markov chains to user session data and devise behavioral patterns.

Chapter 13 studies the problem of community-based semantic subgroup discovery and leverages the structural properties of a complex network, in order to enhance the ontology-based subgroup identification.

We would like to thank all the authors who submitted papers for publication in this book and all the workshop participants and speakers. We are also grateful to the members of the Program Committee and external referee for their excellent work in reviewing submitted and revised contributions with expertise and patience. We would like to thank Hiroshi Motoda for his invited talk on "Which Is More Influential, 'Who' or 'When' for a User to Rate in Online Review Site?" A special thank you is due to both the ECML PKDD Workshop Chairs and to the ECML PKDD organizers who made the event possible. Last but not the least, we thank Alfred Hofmann of Springer for his continuous support.

February 2018

Annalisa Appice
Corrado Loglisci
Giuseppe Manco
Elio Masciari
Zbigniew Ras

Organization

Program Chairs

Annalisa Appice	University of Bari Aldo Moro, Bari, Italy
Corrado Loglisci	University of Bari Aldo Moro, Bari, Italy
Giuseppe Manco	ICAR-CNR, Rende, Italy
Elio Masciari	ICAR-CNR, Rende, Italy
Zbigniew W. Ras	University of North Carolina, Charlotte, USA
	Polish-Japanese Academy of Information Technology, Warsaw, Poland

Program Committee

Martin Atzmueller	Tilburg University, Germany
Elena Bellodi	University of Ferrara, Italy
Petr Berka	University of Economics of Prague, Czech Republic
Claudia Diamantini	Università Politecnica delle Marche, Italy
Hadi Fanaee Tork	University of Oslo, Norway
Bettina Fazzinga	CNR-ICAR, Italy
Filippo Furfaro	Università della Calabria, Italy
Stefano Ferilli	University of Bari, Italy
Dragi Kocev	Jozef Stefan Institute, Slovenia
Gjorgji Madjarov	Ss. Cyril and Methodius University, Macedonia
Mirjana Mazuran	Politecnico di Milano, Italy
Hiroshi Motoda	Osaka University and AFOSR/AOARD, Japan
Amedeo Napoli	LORIA, France
Ruggero G. Pensa	University of Turin, Italy
Gianvito Pio	University of Bari, Italy
Ettore Ritacco	CNR-ICAR, Italy
Samira Shaikh	University of North Carolina, Charlotte, USA
Jerzy Stefanowski	Poznan University of Technology, Poland
Irina Trubitsyna	University of Calabria, Italy
Herna Viktor	University of Ottawa, Canada
Alicja Wieczorkowska	Polish-Japanese Academy of Information Technology, Poland
Wlodek Zadrozny	University of North Carolina, Charlotte, USA

Additional Reviewer

Massimo Guarascio

Which is More Influential, “Who” or “When” for a User to Rate in Online Review Site? (Invited Talk)

Hiroshi Motoda

Osaka University and AFOSR/AOARD, Japan

Abstract. At its heart the act of reviewing is very subjective, but in reality many factors influence user’s decision. This can be called social influence bias. We pick two factors, “Who” and “When” and discuss which factor is more influential when a user posts his/her own rate after reading the past review scores in an online review system. We show that a simple model can learn the factor metric quite efficiently from a vast amount of data that is available in many online review systems and clarify that there is no universal solution and the influential factor depends on each dataset. We use a weighted multinomial generative model that takes account of each user’s influence over other users. We consider two kinds of users: real and virtual, in accordance with the two factors, and assign an influence metric to each. In the former each user has its own metric, but in the latter the metric is assigned to the order of review posting actions (rating). Both metrics are learnable quite efficiently with a few tens of iterations by log-likelihood maximization. Goodness of metric is evaluated by the generalization capability. The proposed method was evaluated and confirmed effective by five review datasets. Different datasets give different results. Some dataset clearly indicates that user influence is more dominant than the order influence while the results are the other way around for some other dataset, and yet other dataset indicates that both factors are not relevant. The third one indicates that the decision is very subjective, i.e., independent of others’ review. We tried to characterize the datasets, but were only partially successful. For datasets where user influence is dominant, we often observe that high metric users have strong positive correlations with three more basic metrics: 1) the number of reviews a user made, 2) the number of the user’s followers who rate the same item, 3) the fraction of the user’s followers who gave the similar rate, but this is not always true. We also observe that the majority of users is normal (average) and there are two small groups of users, each with high metric value and low metric value. Early adopters are not necessarily influential.





Contents

Learning Association Rules for Pharmacogenomic Studies	1
<i>Giuseppe Agapito, Pietro H. Guzzi, and Mario Cannataro</i>	
Segment-Removal Based Stuttered Speech Remediation.	16
<i>Pierre Arbajian, Ayman Hajja, Zbigniew W. Raś, and Alicja A. Wiczorkowska</i>	
Identifying lncRNA-Disease Relationships via Heterogeneous Clustering	35
<i>Emanuele Pio Barracchia, Gianvito Pio, Donato Malerba, and Michelangelo Ceci</i>	
Density Estimators for Positive-Unlabeled Learning.	49
<i>Teresa M. A. Basile, Nicola Di Mauro, Floriana Esposito, Stefano Ferilli, and Antonio Vergari</i>	
Combinatorial Optimization Algorithms to Mine a Sub-Matrix of Maximal Sum.	65
<i>Vincent Branders, Pierre Schaus, and Pierre Dupont</i>	
A Scaled-Correlation Based Approach for Defining and Analyzing Functional Networks	80
<i>Samuel Dolean, Mihaela Dînşoreanu, Raul Cristian Mureşan, Attila Geiszt, Rodica Potolea, and Ioana Ţincaş</i>	
Complex Localization in the Multiple Instance Learning Context	93
<i>Dan-Ovidiu Graur, Răzvan-Alexandru Mariş, Rodica Potolea, Mihaela Dînşoreanu, and Camelia Lemnar</i>	
Integrating a Framework for Discovering Alternative App Stores in a Mobile App Monitoring Platform	107
<i>Massimo Guarascio, Ettore Ritacco, Daniele Biondo, Rocco Mammoliti, and Alessandra Toma</i>	
Usefulness of Unsupervised Ensemble Learning Methods for Time Series Forecasting of Aggregated or Clustered Load.	122
<i>Peter Laurinec and Mária Lucká</i>	
Phenotype Prediction with Semi-supervised Classification Trees	138
<i>Jurica Levatić, Maria Brbić, Tomaž Stepišnik Perdih, Dragi Kocev, Vedrana Vidulin, Tomislav Šmuc, Fran Supek, and Sašo Džeroski</i>	

Structuring the Output Space in Multi-label Classification by Using Feature Ranking	151
<i>Stevanče Nikoloski, Dragi Kocev, and Sašo Džeroski</i>	
Infinite Mixtures of Markov Chains.	167
<i>Jan Reubold, Ahcène Boubekki, Thorsten Strufe, and Ulf Brefeld</i>	
Community-Based Semantic Subgroup Discovery	182
<i>Blaž Škrlj, Jan Kralj, Anže Vavpetič, and Nada Lavrač</i>	
Author Index	197



Learning Association Rules for Pharmacogenomic Studies

Giuseppe Agapito^{1,2} , Pietro H. Guzzi^{1,2} , and Mario Cannataro^{1,2}  

¹ Department of Medical and Surgical Sciences,
University Magna Græcia of Catanzaro, Catanzaro, Italy
{agapito,hguzzi,cannataro}@unicz.it

² Data Analytics Research Centre, University Magna Græcia of Catanzaro,
Catanzaro, Italy

Abstract. The better understanding of variants of the genomes may improve the knowledge on the causes of the individuals' different responses to drugs. The Affymetrix DMET (Drug Metabolizing Enzymes and Transporters) microarray platform offers the possibility to determine the gene variants of a patient and correlate them with drug-dependent adverse events. The analysis of DMET data is a growing research area. Existing approaches span from the use of simple statistical tests to more complex strategies based, for instance, on learning association rules. To support the analysis, we developed GenotypeAnalytics, a RESTful-based software service able to automatically extract association rules from DMET datasets. GenotypeAnalytics is based on an optimised algorithm for learning rules that can outperform general purpose platforms.

Keywords: Association rules · Genomics · SNP

1 Introduction

One of the problems related to drug-development and clinical practice is the variability of the response to the same drug. Pharmacogenomic is a relatively new discipline based on the rationale that this variability is due to different variants in the genome of patients [1]. In particular, it has been shown that a set of genes, defined to as drug absorption, distribution, metabolism and excretion genes (ADME-genes) [2] are related to such processes [3,4]. Such genes present known Single Nucleotide Polymorphisms (SNPs), e.g. variants on the sequence of nucleotides, related to different drug responses [5,6].

To study genome variants, we need: (i) an experimental platform for investigating the presence of SNPs in the ADME genes (among others we consider the Affymetrix DMET platform) [7,8], (ii) a computational platform to associate single or multiple SNPs to drug response. A complete review whose purpose is to highlight the potentiality, reliability, and limitations of the DMET Plus platform as pharmacogenomic drug metabolism multi-gene panel platform for selecting biomarkers is in [9]. In [9] the authors illustrate the possibility to predict

and avoid adverse drug reactions (ADRs), especially in the situation of drugs with a narrow therapeutic index, like antitumor factors. Thus the prediction is of significant relevance in the clinical practice. DMET Affymetrix platform allows investigating germline polymorphisms in a panel of ADME genes, to shed light on the complex relationships between human genetics and drug response and identify new predictive biomarkers to enhance treatment efficacy and safety, a more exhaustive description is available.

Although such analysis is usually performed through statistical analysis, in the following, we will consider analysis approaches based on data mining, i.e. association rule mining. From a computer science point of view, the result of a DMET experiment is a $n \times m$ matrix of alleles, where n is the number of probes ($n = 1936$ in the current DMET plate) and m is the number of samples (patients). Each cell of such table contains a string value including two alleles symbols i.e. a_1/a_2 , where $a_1, a_2 \in \Delta = \{A, C, G, T, -\}$, Δ is the alphabet of allowed symbols, see for instance Table 1 that reports a fragment of DMET data. The alphabet contains only four symbols since, the DNA is composed by nucleotides. Each nucleotide is composed of one of four nitrogen-containing nucleo-bases Adenine (A), Cytosine (C), Guanine (G) and Thymine (T). A probe is a specific region of the human DNA, and for the current version of DMET microarray the DNA is splitted in 1936 different regions of investigation related with the drug response. Subjects (samples or patients) under investigation are arranged in columns. Thus, the i -th column will be represented the i -th subject, and all the alleles (“A/G”, “T/T” and so on) are the detected allele for the i -th patients in each probe.

Table 1. A simple DMET SNP microarray data set. S and P respectively refer to sample and probe identifiers.

Probes	Samples		
	S_1	...	S_N
P_1	G/A	...	T/T
\vdots	\vdots	...	\vdots
P_M	G/A	...	T/C

Usually, the algorithms for the analysis of DMET data try to correlate the presence of genomic variants to the phenotype of patients. Early approaches to the analysis were based mainly on statistical approaches, i.e. DMET-Analyzer [10] employed the well-known Fisher’s Test and several statistical corrections such as Bonferroni or False Discovery Rate.

Although DMET-Analyzer has demonstrated its validity in several clinical studies [3, 4, 7, 11], DMET-Analyzer is not able to cope with multiple variants. To overcome those limitations, we developed DMET-Miner, a novel methodology for the simultaneous analysis of genomic variants in more than one gene.

DMET-Miner employs the association rules mining methodology [12], a well-known method in the data mining field. Despite the innovation introduced by DMET-Miner, it presents some disadvantages due to the Apriori method i.e. the generation of the candidate itemsets could be extremely slow and require a massive amount of main memory [13–15]. CHARM [16] is an efficient algorithm for enumerating the set of all frequent closed itemsets. CHARM breaks the search space into small independent chunks, that can be stored efficiently into main memory making it efficient algorithm to analyze reasonably large databases. CLOSET [17] is an efficient algorithm to mine closed itemsets. CLOSET uses a FP-Tree structure to mine closed itemsets without generate candidate. Closes itemsets are identified developing a single prefix path exploring a partition projection for scalable databases.

To avoid memory issues and to improve the computation of association rules, we here extended the core of DMET-Miner by implementing a modified FP-Growth algorithm able to deal with SNP data efficiently and we implemented it into a new software named GenotypeAnalytics.

The main advantage of FP-Growth concerning Apriori is that FP-Growth does not need to generate a candidate set and it needs to read the input data-set only twice, as opposed to Apriori that reads the input data-set on each iteration. The improvements in terms of memory requirements, as well as computational time introducing by FP-Growth, are evident compared to Apriori as showed in [18]. Moreover we used FP-Growth algorithm because we are not interesting in closed itemsets. Closed itemsets are subsets of frequent itemsets, which are discovered when reducing the learning time is more important then reducing the memory usage. However, limiting the memory usage is not an issue in this study.

GenotypeAnalytics improves the performances of DMET-Miner by using optimised data structures that give good performance results in rule extraction also with massive DMET datasets. Also, GenotypeAnalytics can extract relevant knowledge by computing frequent item-sets efficiently as well as mining association rules [19] that link allelic variants in more than one probe with the health status of patients (e.g. subjects responding or not responding to drugs). The paper is structured as follows: Sect. 2 discusses the related approaches, Sect. 3 introduces the problem, Sect. 4 discusses the proposed algorithm and its implementation, Sect. 5 presents some experimental results, Sect. 6 concludes the paper.

2 Related Work

Existing approaches of analysis of DMET data, span from the preprocessing of raw data, e.g. Affymetrix-power-tools, Affymetrix-DMET-Console, to the variants correlation of different patient conditions, e.g. DMET-Analyzer [10], Cloud4snp [20], coreSNP [13], DMET-Miner [21], and *OS-Analyzer* [22].

- The *apt-dmet-genotype* software of the Affymetrix Power Tools suite, or the DMET Console platform [23], generally allows only the sequential preprocessing of binary data and simple data analysis operations. In particular,

apt-dmet-genotype is a collection of command line programs for analysing and working with Affymetrix microarray data. Whereas, DMET-Console is software tool comes with graphical user interface (GUI) easier to use in order to analyse the intensity data to determine the genotype for each marker. Both programs are generally focused on CEL file analysis. CEL files containing the amount of mRNA produced during the hybridization process.

- DMET-Analyzer [10] is a software platform for the automatic statistical analysis of DMET data that employs the well-known Fisher test and several statistical corrections such as Bonferroni or False Discovery Rate. Although DMET-Analyzer has demonstrated its validity in several clinical studies [3, 4, 7, 11], DMET-Analyzer is not able to cope with multiple variants, and it is not able to group all of them in a single, easy to understand, and biologically relevant information.
- *Cloud4SNP* is the Cloud-based version of *DMET-Analyzer*. *Cloud4SNP* has been implemented on the Cloud using the Data Mining Cloud Framework [24], a software environment for the design and execution of knowledge discovery workflows on the Cloud [25]. *Cloud4SNP* allows to statistically test the significance of the presence of SNPs in two classes of samples using the well known Fisher test. *Cloud4SNP* makes it possible to analyze SNP data sets through a web browser.
- coreSNP is a tool implemented by using Java language, for parallel pre-processing and statistical analysis to cope with high dimensional DMET data sets, deriving from the screening of population. The scalable implementation based on multi-threading allows coreSNP to manage huge volumes of pharmacogenomic experimental. The automatic association analysis among the variation of the patient genomes and the clinical conditions of patients by implementing the well known Fisher’s test, e.g., the different response to drugs.
- *DMET-Miner* is a software tool for the extraction of association rules that correlate the presence of a series of allelic variants with the clinical condition of patients, for example, the combination of alleles typical of a class responsible for the different response to drugs. *DMET-Miner* incorporates a customized SNPs datasets preprocessing approach based on a Fisher’s Test Filter to discard the trivial transactions, decreasing the search space from which to build many independent FP-Tree. *DMET-Miner* thanks to the simple and intuitive graphic user interface, it is a software tool easy to use where specific skills are not necessary to easily extract multiple relations between genomic factors buried into the data sets.
- *OS-Analyzer* is a software tool for the computation and visualization of Overall Survival (OS) and Progression Free Survival (PFS) curves of cancer patients and evaluate their association with ADME gene variants. OS-Analyzer is able to perform an automatic analysis of DMET data enriched with survival events. Moreover, results are ranked according to statistical significance obtained by comparing the area under the curves that is computed by using the log-rank test, allowing a quick and easy analysis and visualization of high-throughput data.

To overcome limitations of the analysis, we developed DMET-Miner, a novel methodology for the simultaneous analysis of genomic variants in more than one gene. DMET-Miner uses on the association rules mining methodology [12], a well-known method in the data mining field. Despite the innovation introduced by DMET-Miner, it presents some disadvantages due to the Apriori method.

3 Problem Statement

DMET datasets are represented as a $m \times n$ SNP DMET table. In particular, m is the number of probes (in the current version of the DMET chip is equals to 1936), whereas n is the number of subjects (patients) gathered for the class of membership i.e. Healthy and Diseased or responding and not responding to the drug. Each element (i, j) of the table contains the allele recognised on the i th probe and at the j th sample. An example of synthetic SNP DMET dataset randomly generate is reported in Table 2.

Table 2. A simple DMET SNP microarray data set. S and P respectively refer to sample and probe identifiers.

Probes	Samples				
	S_1	S_2	S_3	\dots	S_N
P_1	G/A	A/G	A/G	\dots	T/T
P_2	G/A	A/G	A/G	\dots	T/C
\vdots	\vdots	\vdots	\vdots	\dots	\vdots
P_M	G/A	A/G	A/G	\dots	T/C

To extract relevant rules, we need to convert the input DMET data set into a transaction database. The input DMET data needs to be converted in a more suitable data structure to mine association rules. This transformation is mandatory since the data in the current form are not suitable to mine AR. Indeed, it is not possible to treat all the SNPs in the table in the same way because SNPs are different even if have the same name. Only SNPs detected in the same probe can be considered the same. The conversion of DMET data set includes the following steps:

- Loading, Filtering and Transposing the input DMET dataset (let see e.g. Table 2), obtaining a $n \times m$ table of alleles named AllelesTable AT (see Table 3). In this way, each row of the AT contains a transaction and related items. Table 3 shows the transformed matrix AT for the input dataset of Table 2;
- setting-up the desired value minsupport and confidence. Support and confidence are used to filter out rules that have support and confidence lower than the chosen minimum support and confidence respectively;

- Table 3 is then used to extract frequent itemsets;
- Finally the biological interpretation of extracted rules.

Table 3. The AllelesTable AT obtained transposing the input DMET microarray dataset. S and P respectively refer to sample and probe identifiers.

Samples	Probes			
	P_1	P_2	\dots	P_M
S_1	$P_1\text{-G/A}$	$P_2\text{-G/A}$	\dots	$P_M\text{-G/A}$
S_2	$P_1\text{-A/G}$	$P_2\text{-A/G}$	\dots	$P_M\text{-A/G}$
S_3	$P_1\text{-A/G}$	$P_2\text{-A/G}$	\dots	$P_M\text{-A/G}$
\vdots	\vdots	\vdots	\vdots	\vdots
S_N	$P_1\text{-T/T}$	$P_2\text{-T/C}$	\dots	$P_M\text{-T/C}$

To explain the overall process, we here recall some main concepts.

Let $I = \{i_1, i_2, \dots, i_n\}$ be a set of items (alleles), where an item is identified by a specific SNP into a cell (i, j) of AT . Let T be the set of transactions, formally a transaction over I is a couple $T = (tid, I)$, where tid is the transaction identifier, and I is an item or item set. The number of items present in a transaction is defined as *transaction width*. For example, if a transaction comprises 5 items, then its width is equal to 5, whereas a transaction with 10 items has a width of 10. A transaction T_j contains an item set J , if J is a subset of T_j , this is $J \subset T$. Let $D = \{t_1, t_2, \dots, t_m\}$ be a set of transactions, called *DMET-Dataset* D hereafter. Each transaction in D is identified by an unique ID of the corresponding sample or patient.

Now we may start the mining phase by performing the following steps:

1. prune all the items that present a support value lower than the specified minimum frequency threshold.
2. add all the frequent items to the FP-Tree.
3. mine association rules from the FP-Tree.

The power of frequent item sets extraction concerns with the ability to discover interesting relationships hidden in large data sets. This feature relies on a fundamental property of the item set also known as *Support*. Support refers to the number of transactions that contain a particular item or item set. Formally, the Support $S(\cdot)$ of an item X , $S(X)$ can be defined as follows: $S(X) = |\{\forall t_i \subset X \wedge t_i \in T\}|$, where $|\cdot|$ denotes the cardinality of the set. In other words, $S(X)$ is the fraction of transactions in T containing the item/item-set X .

Association models extract rules that express the relationships among items into frequent item sets. We are aimed at finding regularities among SNPs in the DNA, more specifically, to find a set of SNPs that are frequently together that may be related to the disease growth or to the adverse drug reactions. For example, a rule belonging to the frequent item sets composed by the following elements $\{A/A, G/C, C/T\}$, might be stated as: **IF** $(A/A \wedge G/C)$ **THEN** C/T

and, can be read as: if A/A and G/C are included in the transaction, then C/T likely should also be included, since related to the disease under investigation.

4 The Optimised FP-Growth Algorithm of GenotypeAnalytics

This section illustrates the core algorithm of GenotypeAnalytics and its optimizations used to reduce the space search and to minimize the number of mined association rules. The goal of GenotypeAnalytics given a SNPs dataset D is to discover all the frequent patterns above a user support threshold named $min_{support}$ (Minimum Support).

Before converting the input dataset in a transaction database, GenotypeAnalytics tries to reduce the search space through a suitable preprocessing methodology able to decrease the number of possible transactions. The preprocessing method is based on the use of the well known *Fisher's Test* as a filter, which allows removing all statistically insignificant rows. The Fisher's Test makes it possible to evaluate whether the proportions of one variable (SNP) depending on the value of the other variables (SNPs). Thus, Fisher's exact test will tell whether the difference between all the possible couples of SNPs are statistically significant. For example, the null hypothesis is that the probability of becoming cured is the same whether a subject is treated with drug-A or drug-B (for a more exhaustive explanation let consult [21]). Discarding these rows does not lead to lose useful information, but rather allows us to improve the mining of the association rules, avoiding to introduce further biases.

After the filtering step, the resulting table is transformed into a transaction database. Such transformation is necessary since the extraction of frequent item sets is more efficient with this data format. Alleles of different probes sometimes have the same name. Therefore, we modified the variables adding information related to the probe to which each allele belongs using the following notations (i.e. $X.A/A$ and $Y.A/A$, in this example X and Y are two different identifiers of probe, making it possible to distinguish that the two SNPs A/A are different, because detected in two different genomic regions).

The resulting table is stored in a data-structure called *Transaction DB (TDB)* hereinafter). In the TDB the transaction id (TID) is the entry of the table and the matching items set (value) are encoded into hash-set using a hash-function. Thus, it is possible to compress the items as well as, to ensure constant time for standard operations such as: inserting, deleting and searching items in the hash-set.

Despite the preprocessing phase of the input dataset, the number of items that compose the *TDB* is too large to be directly contained into the main memory. Thus, a further compression step is necessary in order to better manage the mining of association rules. For this reason, we decided to implement a customized version of the FP-Growth algorithm, able to deal with SNPs data. The FP-Tree, allows storing the TDB in a compressed form into the main memory

named *FP-Tree*. The *FP-Tree* keeps track of the same item contained in different transactions by connecting the prefix tree nodes indicating the same item into a *frequent items list*. The mining of the associative rules is done using a *Depth-First-Search*, (DFS, in short), sorting in descending order the items in each transaction. The reason behind this choice is that the average size of the conditional *TDB* tends to be smaller if the items are processed in this order. Moreover, the order of the items influences only the search time, not the result of the algorithm.

The GenotypeAnalytics core algorithm needs to scan twice the *TDB*. The first pass is necessary to discover the frequency of each item I into the *TDB* for which $S(I) \geq \text{min}_{sup}$ and sort the items according to their descending frequency. The second *TDB* scan is necessary to delete the items for which the support is $S(I) < \text{min}_{sup}$. Sorting the items in descending order of frequency allows to further compress the *FP-Tree* by limiting the number of different possible prefixes. Now all the items into the *TDB* can be mapped on the *FP-Tree*.

The mapping is performed by means of *support-update* and *node-creation* functions. If during the mapping, the current element in the transaction matches the current element in the *FP-Tree* the function *support-update*, which updates the support of the current node, is invoked (increasing the occurrence of the current node). Whereas, if the current node in the *FP-Tree* and the current node in the transaction do not match the function *node-creation* is called. The *node-creation* function starting from the current item creates a new node, adding it as children of the current *FP-Tree* node. The other items in the current transaction are appended as children of the last created *FP-Tree* node, as shown in Fig. 1. The creation of *FP-TREE* is an iteratively process that ends when there are no more transactions contained into the *TDB*.

Finally, association rules are mined by using a recursive methodology of Tree visiting, in particular, we defined an *inverted DFS (Deep First Search)* scan method to explore the *FP-Tree*. Algorithm 1 reports the pseudo code of the GenotypeAnalyzer core algorithm.

5 Performance Evaluation

In this section, we present the performance evaluation of our version of the *FPGrowth* algorithm compared to the *FP-Growth* algorithm available in *SPMF*, an Open-Source Data Mining Library [26], and the version proposed in [27]. Experiments have been ran on the same data sets, namely “*Vote.arff*”, “*Supermarket.arff*” and a synthetic “*DMET-SNP*” data set. Table 4 reports the characteristics of the three datasets. To provide an unbiased comparison, we employed biological and non-biological datasets as a benchmark to test the association rule mining capability of each tool. In this way, we can highlight the capability of GenotypeAnalytics to mine association rules by analysing biological datasets as well as by analysing general datasets. We have to use a synthetic *DMET-SNP* dataset because at the moment of writing a real dataset was not be available.

id_1	AM_3-A/A, AM_4-A/G, AM_10-T/T
id_2	AM_11-A/A, AM_3-AG, AM_10-T/T

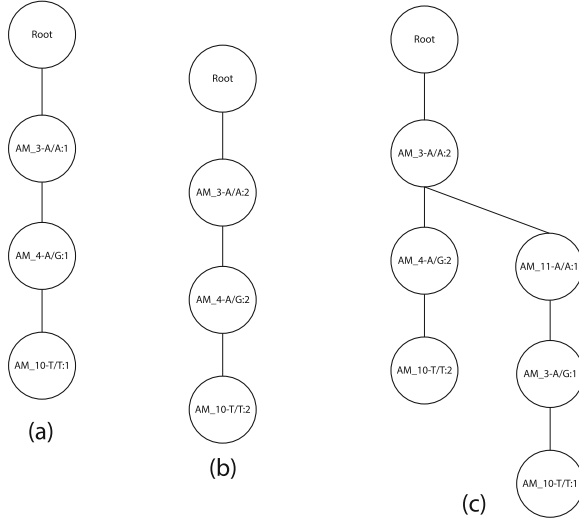


Fig. 1. Support-update and node-creation functions. When the item $AM_3 - A/A$ in transaction id_1 is added to the tree, it is necessary to just update its support by increasing it through the support-update function. support-update is invoked for all the remainder items in this transaction (let see Fig. 1(a) and (b)). When item $AM_{11} - A/A$ has to be added, the function node-creation is invoked because there is not match between the current item and the current node in the tree; any other item in transaction id_2 is added as child of the node $AM_{11} - A/A$ (let see Fig. 1(c)).

Table 4. Datasets used for the experiments.

Dataset name	Number of rows	Number of attributes
Vote	435	17
Supermarket	4627	217
DMET	1936	1000

A real DMET-SNP dataset has to be prepared in wet-laboratory by the clinical unit after collected human samples from patients enrolled in a study and analyzed by microarray.

As proof-of-principle, we report the performance evaluation results of all the FP-Growth implementations. All the experiments have been executed on a machine equipped with a Pentium i7 2.5 GHz CPU, 16 GB RAM and a 512 GB SSD disk. The reported execution times refer to average times; each value has

Algorithm 1. FP-Growth Core Algorithm.

Require: SNP DMET data set D

```

1: for all rows  $\in T'$  do
2:   if (rowsDistribution  $\geq$  Thr) then
3:     pvalue  $\leftarrow$  computeFisherTest(row)
4:     if (pvalue  $>$   $\sigma_{Thr}$ ) then
5:       remove(row)
6:     else
7:       TDB  $\leftarrow$  convert(row)
8:     end if
9:   end if
10: end for

11: for all items  $\in$  TDB do
12:   if item.freq  $\leq$  minsupp then
13:     TDB.remove(item)
14:   else
15:     update(frequentItemsList)
16:   end if
17: end for
18: FPTree.add(frequentItemsList)
19: for all t  $\in$  TDB do
20:   if t  $\in$  header of FPTree then
21:     supportUpdate
22:   else
23:     nodeCreation
24:   end if
25: end for

26: for all (item  $\in$  header of FPTree) do
27:   cpb  $\leftarrow$  generate(item, FPTree)
28:   while cpb =  $\emptyset$  do
29:     if cpbnodefreq  $<$  minsupp then
30:       remove(cpbnode)
31:     end if
32:   end while
33: end for

```

been computed repeating 10 times the experiments with the same settings. In this way, it is possible to ensure that the results are comparable.

Figures 2, 3 and 4 convey the execution times obtained analyzing the data sets by the three different implementation of the FP-Growth algorithms. All the execution times are obtained by varying the minimum support values. The solid black line refers to our implementation of the FP-Growth algorithm, the dashed green line refers to the FP-Growth version available in *SPMF*, and finally the dash-dot red line refers to the FP-Growth version proposed in [27].

All these implementations of FP-Growth show good performance on the classical Vote and Supermarket data sets. Our implementation of FP-Growth does

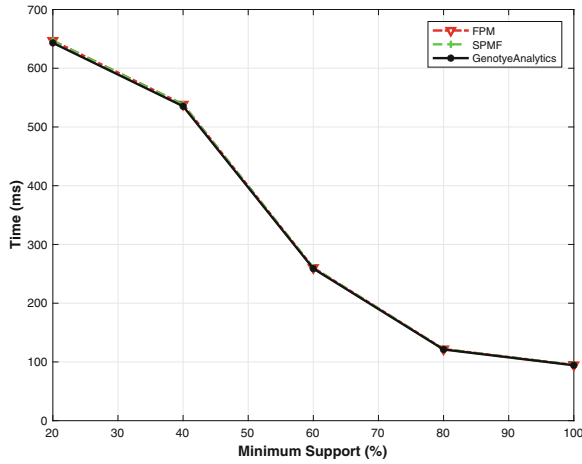


Fig. 2. Execution times of various implementations of FP-Growth algorithm on the Vote data set. The execution times are obtained by varying the value of minimum support. (Color figure online)

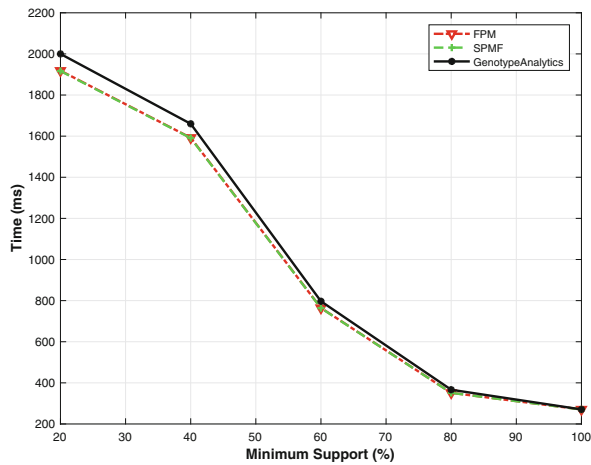


Fig. 3. Execution times of various implementations of FP-Growth algorithm on the Supermarket data set. The execution times are obtained by varying the value of minimum support. (Color figure online)

not present many differences compared to the other two methods, showing very similar performance to those of other tools, with the exception of the synthetic SNP data set. Here the results of the other two methods were beaten with an appreciable margin by our implementation of FP-Growth, as our version of FP-Growth is highly optimized to dig with SNPs data, thus clearly performs best.

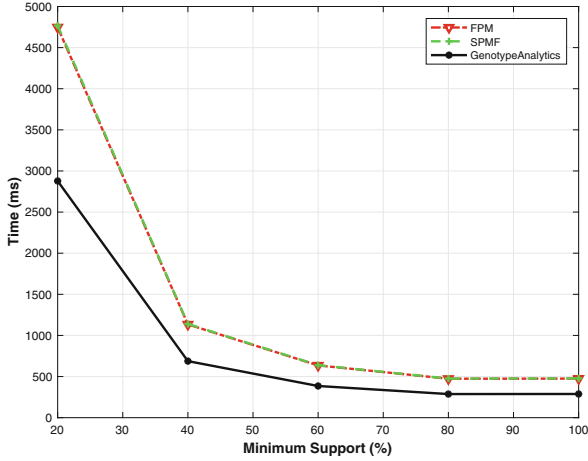


Fig. 4. Execution times of various implementations of FP-Growth algorithm on the DMET-SNP data set. The execution times are obtained by varying the value of minimum support. (Color figure online)

A rules mined by using GenotypeAnalytics is:

- AM_13459_T/C:34, AM_13464_C/A:34, AM_13465_C/T:35;

After translating the rule by using the GenotypeAnalytics probe converter function, it is easier to analyze the mined rule.

Table 5. Translated rule.

Probe ID	DMET detected allele	Gene name	Chromosome	Functional consequence
AM_13459	T/C	<i>UGT2B7</i>	rs7668258	<i>intron variant, upstream variant 2KB</i>
AM_13464	C/A	<i>UGT2B7</i>	rs7438284	<i>synonymous codon</i>
AM_13465	C/T	<i>UGT2B7</i>	rs7439366	<i>missense</i>

A possible approach to explain the rule extracted from the synthetic dataset could be the following. In the translated rule (see Table 5); it is worthy to note that all mutations refer to the same chromosomes, affect the following functional consequences. The *intron variant* in the chromosome “rs7668258” is due to the presence of “T/C” SNP. Intronic variants might affect alternative splicing of the *mRNA*, or change the level of gene expression. Upstream variant refers to the *transcription direction*, by convention upstream is toward the 5′ end of the coding strand. The “C/A” allelic variant into the chromosome rs7438284 affects the synonymous codon functionality. Synonymous codon functionality defines

the insertion of the same amino acid, differing in its decoding characteristics. Finally, an allelic variant in the chromosome *rs7439366* might be produce a missense in a codon, coding for a different amino acid. Thus researchers could spur easily light in the dynamics that govern the hidden interactions among genes, governing the cellular cycle.

6 Conclusion

Analysing genotyping datasets presents various challenges due to the huge volumes of data and due to the specific characteristics of SNPs data. Thus, using general purpose data mining implementation is not feasible and for this reason we implemented GenotypeAnalytics, a specialised association rule mining system able to mine association rules from DMET genotype data. It includes an optimised version of the implementation of the FP-Growth algorithm. Preliminary experiments show how our solution outperforms off the shelf implementation of FP-Growth. As well as, GenotypeAnalytics makes easy to analyse SNP datasets without to be necessary to have powerful computers that are expensive and require a lot of maintenance to work properly. Moreover, the association rules provided by GenotypeAnalytics are easy to read and understand, highlighting in a remarkable way for the researchers, which are the multiple variants associate to the particular condition under investigation. As future work, we will investigate automatic methods to rank the extracted rules on the basis of their biological significance and memory by using real DMET dataset in the oncology field.

Acknowledgements. This work has been partially funded by the following research projects:

- “BA2Know-Business Analytics to Know” (PON03PE_00001_1), funded by the Italian Ministry of Education and Research (MIUR)
- INdAM - GNCS Project 2017: “Efficient Algorithms and Techniques for the Organization, Management and Analysis of Biological Big Data”.

References

1. Meyer, U.A.: Pharmacogenetics and adverse drug reactions. *Lancet* **356**(9242), 1667–1671 (2000)
2. Li, J., Zhang, L., Zhou, H., Stoneking, M., Tang, K.: Global patterns of genetic diversity and signals of natural selection for human ADME genes. *Hum. Mol. Genet.* **20**(3), 528–540 (2011)
3. Lombardi, G., Rumiato, E., Bertorelle, R., Saggiaro, D., Farina, P., Della Puppa, A., Zustovich, F., Berti, F., Sacchetto, V., Marcato, R., et al.: Clinical and genetic factors associated with severe hematological toxicity in Glioblastoma patients during Radiation Plus Temozolomide treatment: a prospective study. *Am. J. Clin. Oncol.* **10**, 1097 (2013)

4. Di Martino, M.T., Arbitrio, M., Guzzi, P.H., Leone, E., Baudi, F., Piro, E., Prantera, T., Cucinotto, I., Calimeri, T., Rossi, M., Veltri, P., Cannataro, M., Tagliaferri, P., Tassone, P.: A peroxisome proliferator-activated receptor gamma (PPARG) polymorphism is associated with Zoledronic acid-related Osteonecrosis of the jaw in multiple Myeloma patients: analysis by DMET microarray profiling. *Br. J. Haematol.* **154**, 529–533 (2011)
5. Guzzi, P.H., Agapito, G., Milano, M., Cannataro, M.: Methodologies and experimental platforms for generating and analysing microarray and mass spectrometry-based omics data to support P4 medicine. *Briefings Bioinf.* **17**(4), 553–561 (2015)
6. Arbitrio, M., Di Martino, M.T., Barbieri, V., Agapito, G., Guzzi, P.H., Botta, C., Iuliano, E., Scionti, F., Altomare, E., Codispoti, S., et al.: Identification of polymorphic variants associated with erlotinib-related skin toxicity in advanced non-small cell lung cancer patients by DMET microarray analysis. *Cancer Chemother. Pharmacol.* **77**(1), 205–209 (2016)
7. Di Martino, M.T., Arbitrio, M., Leone, E., Guzzi, P.H., Saveria Rotundo, M., Ciliberto, D., Tomaino, V., Fabiani, F., Talarico, D., Sperlongano, P., Doldo, P., Cannataro, M., Caraglia, M., Tassone, P., Tagliaferri, P.: Single nucleotide polymorphisms of ABCC5 and ABCG1 transporter genes correlate to irinotecan-associated gastrointestinal toxicity in colorectal cancer patients: a DMET microarray profiling study. *Cancer Biol. Ther.* **12**(9), 780–787 (2011)
8. Guzzi, P.H., Cannataro, M.: μ -CS: an extension of the TM4 platform to manage Affymetrix binary data. *BMC Bioinform.* **11**(1), 315 (2010)
9. Arbitrio, M., Di Martino, M.T., Scionti, F., Agapito, G., Guzzi, P.H., Cannataro, M., Tassone, P., Tagliaferri, P.: DMETTM (Drug Metabolism Enzymes and Transporters): a pharmacogenomic platform for precision medicine. *Oncotarget* **7**(33), 54028 (2016)
10. Guzzi, P., Agapito, G., Di Martino, M., Arbitrio, M., Tassone, P., Tagliaferri, P., Cannataro, M.: DMET-analyzer: automatic analysis of Affymetrix DMET data. *BMC Bioinform.* **13**(1), 258 (2012)
11. Rumiato, E., Boldrin, E., Amadori, A., Saggioro, D.: DMET (Drug-Metabolizing Enzymes and Transporters) microarray analysis of colorectal cancer patients with severe 5-fluorouraci-induced toxicity. *Cancer Chemother. Pharmacol.* **72**(2), 483–488 (2013)
12. Agrawal, R., Imieliński, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases, vol. 22, pp. 207–216. ACM, New York (1993)
13. Guzzi, P.H., Agapito, G., Cannataro, M.: coreSNP: parallel processing of microarray data. *IEEE Trans. Comput.* **63**(12), 2961–2974 (2014)
14. Di Martino, M.T., Guzzi, P.H., Caracciolo, D., Agnelli, L., Neri, A., Walker, B.A., Morgan, G.J., Cannataro, M., Tassone, P., Tagliaferri, P.: Integrated analysis of microRNAs, transcription factors and target genes expression discloses a specific molecular architecture of hyperdiploid multiple myeloma. *Oncotarget* **6**(22), 19132 (2015)
15. Di Martino, M.T., Scionti, F., Sestito, S., Nicoletti, A., Arbitrio, M., Guzzi, P.H., Talarico, V., Altomare, F., Sanseviero, M.T., Agapito, G., et al.: Genetic variants associated with gastrointestinal symptoms in fabry disease. *Oncotarget* **7**(52), 85895 (2016)
16. Zaki, M.J., Hsiao, C.J.: CHARM: an efficient algorithm for closed itemset mining. In: *Proceedings of the 2002 SIAM International Conference on Data Mining*. SIAM, pp. 457–473 (2002)

17. Pei, J., Han, J., Mao, R., et al.: CLOSET: an efficient algorithm for mining frequent closed itemsets. In: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, vol. 4, pp. 21–30 (2000)
18. Agapito, G., Milano, M., Guzzi, P.H., Cannataro, M.: Extracting cross-ontology weighted association rules from gene ontology annotations. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **13**(2), 197–208 (2016)
19. Agapito, G., Cannataro, M., Guzzi, P.H., Milano, M.: Using go-war for mining cross-ontology weighted association rules. *Comput. Methods Programs Biomed.* **120**(2), 113–122 (2015)
20. Agapito, G., Cannataro, M., Guzzi, P.H., Marozzo, F., Talia, D., Trunfio, P.: Cloud4SNP: distributed analysis of SNP microarray data on the cloud. In: Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics, p. 468. ACM (2013)
21. Agapito, G., Guzzi, P.H., Cannataro, M.: DMET-Miner: Efficient discovery of association rules from pharmacogenomic data. *Journal of biomedical informatics* **56**, 273–283 (2015)
22. Agapito, G., Botta, C., Guzzi, P.H., Arbitrio, M., Di Martino, M.T., Tassone, P., Tagliaferri, P., Cannataro, M.: OSAalyzer: a bioinformatics tool for the analysis of gene polymorphisms enriched with clinical outcomes. *Microarrays* **5**(4), 24 (2016)
23. Sissung, T., English, B., Venzon, D., Figg, W., Deeken, J.: Clinical pharmacology and pharmacogenetics in a genomics era: the DMET platform. *Pharmacogenomics* **11**, 89–103 (2010)
24. Marozzo, F., Talia, D., Trunfio, P.: A cloud framework for big data analytics workflows on Azure. In: Cloud Computing and Big Data. *Advances in Parallel Computing*, vol. 23, pp. 182–191. IOS Press (2013). <https://doi.org/10.3233/978-1-61499-322-3-182>
25. Marozzo, F., Talia, D., Trunfio, P.: Using clouds for scalable knowledge discovery applications. In: Caragiannis, I., Alexander, M., Badia, R.M., Cannataro, M., Costan, A., Danelutto, M., Desprez, F., Krammer, B., Sahuquillo, J., Scott, S.L., Weidendorfer, J. (eds.) Euro-Par 2012. LNCS, vol. 7640, pp. 220–227. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36949-0_25
26. Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.W., Tseng, V.S.: SPMF: a java open-source pattern mining library. *J. Mach. Learn. Res.* **15**(1), 3389–3393 (2014)
27. Borgelt, C.: Frequent item set mining. *Wiley Interdisc. Rev. Data Mining Knowl. Discovery* **2**(6), 437–456 (2012)



Segment-Removal Based Stuttered Speech Remediation

Pierre Arbajian¹(✉), Ayman Hajja², Zbigniew W. Raś^{1,3,4},
and Alicja A. Wieczorkowska³

¹ Department of Computer Science, University of North Carolina,
9201 University City Blvd., Charlotte, NC 28223, USA
{arbajian,ras}@unc.edu

² Department of Computer Science, College of Charleston,
66 George Street, Charleston, SC 29424, USA
hajjaa@cofc.edu

³ Multimedia Department, Polish-Japanese Academy of Information Technology,
Koszykowa 86, 02-008 Warsaw, Poland
alicja@poljap.edu.pl

⁴ Institute of Computer Science, Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warsaw, Poland

Abstract. Speech remediation by identifying those segments which take away from the substance of the speech content can be performed by correctly identifying portions of speech which can be deleted without diminishing from the speech quality, but rather Pierre improving the speech. Speech remediation is especially important when the speech is disfluent as in the case of stuttered speech. In this paper, we describe a stuttered speech remediation approach based on the identification of those segments of speech which when removed would enhance speech understandability in terms of both speech content and speech flow. The approach we adopted consists of first identifying and extracting speech segments that have weak significance due to their low relative intensity, then classifying the segments that should be removed. We trained several classifiers using a large set of inherent and derived features extracted from the audio segments for the purpose of automatic improvement of stuttered speech by providing a second layer filtering stage. This second layer would discern the audio segments that need to be eliminated from the ones that do not. The resulting speech is then compared to the manually-labeled “gold standard” optimal speech. The quality comparisons of the resulting enhanced speeches and their manually-labeled counterparts were favorable and the corresponding tabulated results are presented below. To further enhance the quality of the classifiers we adopted a voting techniques that encompassed an extended set of models from 14 algorithms and presented the classifier performance measures from different voting threshold values. This voting approach allowed us to improve the specificity of the classification by reducing the false positive classifications at the expense on additional false negatives thus improving the practical effectiveness of the system.

Keywords: Stuttering detection · Speech analysis
Speech remediation · Classification

1 Introduction and Background

Quality of life is negatively impacted when individuals are chronically unable to express themselves due to lack of speech fluency. Stuttering, also referred to as stammering, is a condition that is exhibited in bad fluency of speech. The onset of stuttering generally occurs during childhood years, and in one fourth of cases this speech impediment persists throughout life [8, 11].

Automatic speech disfluency detection offers many advantages, ranging from time saving, constant speech monitoring, reducing the subjectivity of manual disfluency identification [5, 9] as well as a more effective automatic speech recognition. Therefore, the identification of “episodes” of stutter will offer firm and actionable information [3–5]. Reliable identification of speech segments, from the beginning of episode to the end of episode, with pauses, blocks, interjections and hesitations will allow speech cleanup by ridding the speech of the blocks, and interjections, smoothing the hesitation segments and shortening the prolongations [6, 7, 10].

The state of the art is rich with papers which describe various prosodic- and semantic-based machine learning approaches and algorithms for the detection of disfluent speech [1, 9, 11–13]; however, research work addressing disfluent speech remediation is considerably less common. Stuttered speech is difficult to listen to; our assumption in this research work is that speech clarity will be significantly improved by eliminating those segments. As a result the listener will more intent on listening to the speech.

To confirm the assumption that undesired speech segment removal enhances speech quality, we ran a script that eliminates audio segments with low intensity levels using varying intensity thresholds and segment durations. This resulted in an unquestionably clearer and more intelligible audio compared to the original speech. That being said, automatically detecting and removing these *potentially undesired* segments, resulted occasionally in discarding the wrong segments. For that reason, we introduced a second layer of filtering stage when the *potentially undesired* segments are further classified into *true undesired*, which would need to be discarded; and *false undesired*, which would need to be retained.

The complexity in this paper arose from the need to take into consideration various overlapping multi-threshold data samples. Also, we devised a labeling process which reduces the effort required during the labeling process. We feel that the proposed adaptive data collection and processing method paves the way and lays a sound foundation for successful treatment of future and similar multi-perspective [15] data collection and classification challenges.

In this paper, we discuss one certain type of stuttered speech remediation; remediation in the form of undesired speech segments removal. In the next section, we present an algorithm to enhance the stuttered speech quality by eliminating the speech segments that contain stuttering *blocks*, which are defined as

the segments [14] of audio in which the person who stutters is unable to produce clear sounds. Furthermore, unwanted segment elimination provides the listener with yet another benefit; that is, the speech is shortened (up to 60% reduction) without speeding up individual syllables.

Eliminating disfluent segments of speech requires identifying those segments which do not add any value to the speaker’s message. Our algorithm first identifies and creates sound files of those segments that have the distinct potential of being undesired in the speech through linguistic intensity and length analysis; detailed explanation will be provided in Sect. 2. Next, the set of *potentially undesired* audio segments are listened to in order to determine whether they have semantic value or not, and to determine whether they must be deleted or retained accordingly. The final output of the speech audio file, after removing the segments that were manually-labeled *true undesired*, will represent the “gold standard”. The “gold standard” speech is later used to measure the performance of the trained classifier. There are two vital reasons for the phase in which the *potentially undesired* segments are manually labeled; the first reason is to generate the optimal speech that will be used to evaluate our enhanced stuttered speeches, and the second reason is to generate the *true undesired* and *false undesired* segments used to build our classifier.

The dataset of speeches that we used was obtained from the UCLASS series of recordings. UCLASS, which stands for the University College London Archive of Stuttered Speech, is a database of stuttered speech recordings with individual speech and speaker metadata. UCLASS contains a collection of over one hundred speeches from which we chose fifteen for our research study.

The approach that we followed was to begin by scanning each speech for potentially stuttered segments according to varying intensity thresholds and lengths of audio segments. Every *potentially undesired* extracted segment is considered a candidate for removal because of the selection criteria used; however, in numerous cases, the candidate segment may contain semantic speech and therefore should not be omitted from the original speech. Determining which candidate segments are truly undesired candidates and which segments are not, is done through a classification system that was trained by providing it with manually-labeled *true undesired* and *false undesired* segments that were collected beforehand.

The features that are used in the classifier training include data extracted from both the frequency domain such as voice formants and pitch, and from the time domain such as intensity. In addition to that, a combination of speaker and speech metadata are also used to improve the trained classifier; a more elaborate description of the list of features will be presented in the next section.

The difference between the quality of the “gold standard” of a given speech compared to the quality of an enhanced speech after having every *potentially undesired* segment being classified is used to evaluate the quality of our system. The results with respect to classifier performance are tabulated in the findings section according to the extraction parameters. Process efficiency improvements were created to minimize manual labeling effort as well as segment deletion redundancy.

As we remove speech portions we must be careful to avoid deleting good content i.e. we are quite reluctant to remove segments carrying meaning. We consider such mistakes more detrimental to the speech than not removing blocked voice portions. In order to minimize the rate of mistaken speech segment removal we doubled the number of classifiers and performed the classification according to a threshold vote cutoff. Instead of a majority vote we selected multiple voting proportion cutoffs, tabulated and graphed the outcome for the respective True Positives, True Negatives, False Positives, False Negatives, Accuracy, Specificity, and True Positive Rates. This exercise was instrumental in helping us decide on a voting threshold which greatly reduces the number of wrong segment omissions at a reasonable rate of increase in non removal of blocked speech segments.

2 Proposed Method

The method we adopted for this research consisted of selecting a set of speeches to be examined from a group of disfluent speeches available from UCLASS. UCLASS speeches exhibit a wide variety of speakers and stuttering conditions, yet the vast majority of the speeches are distinctly stuttered and quite disfluent. We selected fifteen speeches (roughly one hour of stuttered speech) as the foundation for our research. Our goal in this research is to develop a system capable of automatically analyzing stuttered speeches for the purpose of detecting undesired segments and enhancing the overall speech quality through eliminating the disfluent segments. In this work, we used a *Praat* script for initially detecting potential candidate segment to be removed from the stuttered speech; the ultimate goal of this research is to build a system able to distinguish *true undesired* from *false undesired* for all candidate segments. Next, we provide an elaborate description of the workings of the overall process.

2.1 Extracting Potentially Undesired Candidate Segments

Fifteen speeches from the UCLASS repository are selected as the stuttered speeches of interest, each is scanned with a total of 8 segment extraction parameters. The extraction parameters determine what the sound intensity threshold and the minimum duration of those segments should be. The minimum durations that we used to iterate through our speeches are 0.6, 0.8, 1.0 and 1.2 s, and the speech intensity thresholds are at 95% and 90% of the entire speech intensity in decibels; for example, if the average intensity of a given speech is 50 db, then the intensity thresholds that correspond to 95% and 90% are 47.5 db and 45 db, respectively.

For each possible pair combination of segment duration and intensity threshold (total of 8 unique combinations), the entire speech is scanned and the *potentially undesired* candidates are detected according to the corresponding pair combination. The minimum segment duration ensures that no excessively short segments are extracted as potential candidates for deletion, which would disrupt

the flow of normal speech and unfavorably affect the speech cadence. Note that the extracted segments from each unique pair will not overlap, and that only such overlapping is possible, and rather likely to happen, when examining segments extracted from different pairs. We provide a demonstration of how the segments may look like in Fig. 1. For example, according to Fig. 1, extracting potentially undesired segments from the fourth row (0.6 minimum duration with 95% intensity threshold) will result in three different segments (*B*, *G*, and *K*); since segment *B*'s length is 0.8 s, which satisfies the minimum duration threshold for the third row, then the same segment will also be identified as a *potentially undesired* segment on the third row; however, since the length of the segment is not 1.0 s (or more), that segment will not be identified on the second (or first) row in Fig. 1.

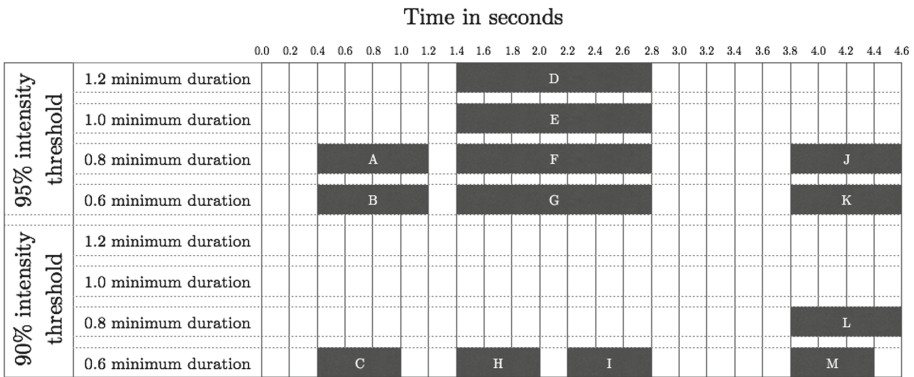


Fig. 1. Extracting potentially undesired candidate segments

Each extracted candidate segment results in a waveform audio file accompanied by five additional vectors that serve as the segment inherent features, three out of which contain the values of formant 1, formant 2, and formant 3 and the remaining two vectors contain the pitch values and the intensity values. All formants were computed with a 50 ms Gaussian analysis window with a 12.5 ms offset. The pitch values were taken at 10 ms interval and computed with the *Praat* software algorithm which performs an acoustic periodicity detection on the basis of an accurate autocorrelation method as described in [2]. The intensity values were computed by squaring then convolving the sound values within Gaussian analysis window of 50 ms length. The waveform audio files will be later used for manually labeling the candidate segment, and the set of inherent features (in addition to a set of derived features presented in Subsection 2.3) will be used to build our classifier.

2.2 Labeling Potentially Undesired Segments

Following the generation of the segment files, the labeling process is performed to begin building the classifier and testing our dataset. The segment candidates must be manually labeled as: (1) delete and analyze, these are denoted by G, (2) leave in the speech and analyze, these are denoted by B, and (3) no analysis, but delete from the speech, these segments of speech represent external sounds such as when the interviewer was attempting to speak softly. We chose to exclude these segments from the speech during cleanup because they showed no speech disfluency but refrained from including them in the training dataset.

The “delete and analyze” labels (G) indicate that the sound is void of semantic meaning and that we would like for our classifier to mark such segments as “delete”, or as *true undesired*. The “leave in the speech and analyze” label is used for sounds that were long enough and with low intensity yet the segment sound contained semantic meaning and must not be removed from the original speech; our classifier should label such segment as “retain”, or *false undesired*. The “delete and do not analyze” label indicates that a segment is best deleted from the speech but does not represent the type of sound that our classifier must learn to recognize or take any action about; an common example would be a segment containing the soft voice of the interviewer.

In an attempt to avoid redundant labeling efforts and minimize the number of segments which must be manually reviewed, we have sorted the list of segments to be reviewed such that if a segment s is marked “delete”, then all other segments that are contained within s (start with, or after, s ; and end when s ends, or anytime before that) will also be marked as “delete”. The reasoning behind our approach is that if some segment s is void of semantic meaning, then any other segment contained in s must also be void of semantic meaning. Note here that it is not the case that the opposite is true; given that some segment s contains some semantic meaning, we cannot conclude that all other segments contained in s must also contain semantic meaning. This approach has allowed us to reduce review time by a factor of five.

We will use Fig. 1 to demonstrate the process explained above. The first step is to sort all *potentially undesired* segments according to their start time, from top to bottom. This means that based on Fig. 1, the list of candidate segments will be sorted as follows: $A, B, C, D, E, F, G, H, I, J, K, L$, and M . The first candidate segment to be listened to is A ; if the segment A is marked as *true undesired* (no semantic value), then all other subsets that are contained in A are also marked *true undesired*, which are B and C . Similarly, if segment D is labeled as *true undesired*, then segments E, F, G, H , and I are also labeled *true undesired*. If however, segment D was marked as *false undesired* (contains semantic value), then this only implies that segments E, F , and G are also *false undesired* (since they are essentially the same as segment D), but we cannot conclude that segment H nor I are also *false undesired*; therefore, we would need to listen to the two segments H and I to determine what the label of each of them should be.

The dataset used in our classification exhibited two possible class values: G (positive), which meant that the candidate sound must be deleted; and B (negative), which meant that the sound segment should not be deleted.

2.3 Building the Classifiers

In addition to the label feature, the dataset used to train our classifiers will contain two additional types of features: (1) a set of derived attributes extrapolated from the set inherent features, and (2) speaker and speech metadata [15]. The inherent features that are associated with each candidate segment are formant 1, formant 2, and formant 3 (sampled at 50 ms intervals with the time step being 12.5 ms), pitch (sampled at 10 ms), and intensity (sampled at 8 ms). The derived features that are used in our classifier training are all extrapolated from the inherent features and will serve as a set of data-points that provide information about the entire segment as opposed to an exact point in time (due to sampling). We start by calculating the derivatives of each one of the five inherent features; the reason for calculating the derivatives is to assess the variance of our inherent features, which is vital for detecting stutter. Then, for each one of the inherent features and their derivatives, we calculate the average, median, standard deviation, percentiles (25, 50, and 75), minimum value, maximum value, peak-to-peak amplitude measurement, and variance.

The speaker and speech metadata we included in our dataset consisted of those data features provided with the UCLASS dataset; they consisted of the following:

Speaker category metadata: Gender (M/F), Handedness (L, R, not known), Past history of stuttering in the family, Age of stuttering onset, Age at the time of recording, Location of recording (clinic, UCL, or home), Recording conditions (quiet room or sound-treated room), Type of therapy received (family based treatment or holistic treatment), Time between therapy and recording time, Speaker had any history of hearing problems (Y/N), Speaker had a history of language problems (Y/N), and Special educational needs (Y/N).

Speech category metadata: Background acoustic noise level (numeric), environmental noise level (numeric), speaker clarity (numeric), interlocutor intrusiveness (numeric).

The resulting dataset consisting of signal statistics, metadata, and labels consisted of 126 features and was used to train eight different classifiers, each for a unique pair combination of minimum duration and intensity threshold (see Algorithm 1). Recall that the minimum duration threshold is the minimum length of *potentially undesired* segments, and that the intensity (percentage) threshold is measured by examining the averaged speech intensity. We used the *R* ‘caret’ package in order to streamline the classifier testing process and cover a wide range of classifiers.

Algorithm 1. Summary of our proposed algorithm

```

for every segment duration and intensity threshold pair do
  | extract potentially undesired candidate segments from a given speech;
  | manually label each candidate as true undesired or false undesired;
end
training and testing phase;
for every segment duration and intensity threshold pair do
  | train a classifier using a portion of the labeled candidate segments;
  | test the remaining segments using the classifier built above;
  | evaluate the resulting labels by comparing them to the “gold
  | standard”;
end

```

There are three different courses of action for segments removal that are examined in this research work, each producing a different level of stutter remediation:

1. Remove all candidate segments without human intervention (this approach does not require manual labeling).
2. Remove only the candidate segments that are manually labeled *true undesired* (resulting speech is referred to by the “gold standard”).
3. Classify all candidate segments using our trained classifier, and only remove the candidates that are classified *true undesired*.

Because of the speech intensity weakness during the episodes to be deleted we were able to remove the segment minus 50 ms from the front both the front and end of the episode without creating any perceptible sound discontinuities.

Each removal course of action yields a certain “enhanced” speech file; the enhanced files are then examined and compared. Our goal is to build a classifier that generates an audio file (course of action number 3) that is as close as possible to the golden standard audio file (generated from course of action number 2).

In the case of speech remediation, it is more important to avoid deleting semantically meaningful segments than to miss deleting semantically meaningless segments. In other terms, when evaluating our classifier performance, false negatives (deleting what should not be deleted) should be considered more undesirable than false positives (missing to delete segments which must be deleted). The R classifiers that we trained and tested are C5.0, Neural Networks, Recursive Partitioning, Random Forests, Polynomial SVM, and AdaBoost. In Fig. 2 below we show the process flow starting at the top left *Extraction* of the candidate segments followed by two independent paths (1) *Feature* extraction and (2) *Labeling*; the results of the two flows provides the digital signal processing predictors and manual label results which will be combined with the *metadata* to create the complete dataset.

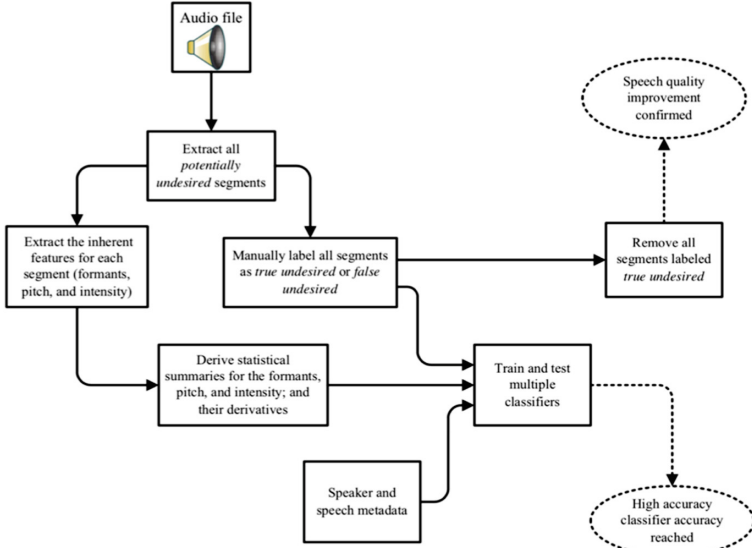


Fig. 2. Depiction of our proposed method

2.4 Practical Classifier Enhancement

We mentioned the impact of removing *block* episodes throughout the paper and established the detrimental effect of removing episodes which are not truly bad. In this section, we will revisit the segment removal concept as it pertains to the optimization of speech repair.

The benefits of the identification and removal of *blocked* segments is multi-fold; the listener benefits in a multitude of ways:

1. He/she does not hear spurious sounds which provide no meaningful lexical content to the recording.
2. The flow of the recording is conducive to a smoother listening experience because of increased fluency.
3. The speech is shortened thus reducing speech length.

The speaker benefits also, we list two such advantages:

1. By listening to the repaired speech, he/she acquires the confidence that some prosodic speech repairs will greatly improve the message delivery.
2. The speaker can listen to the message on the merit of its meaning instead of style thereby focusing on the core semantic components of the recording

Original, pre-repair speeches are qualitatively weak, yet their prosodic and lexical content is uncompromised. If sought indiscriminately, the benefits listed above could compromise the speech where meaning loss is incurred due to the repair process.

The aim of the described system enhancement is to generate a speech which is superior in practical usability without the collateral damage of lexical component resulting from removing segments that were not supposed to be removed.

The reason for remedy optimization is to ensure that as little meaning loss as possible is caused thanks to the intervention from our proposed process. To this end, we remain cognizant that meaning loss is more “detrimental” to the message conveyed by a recording than the inconvenience of hearing unnecessary *blocks* of speech. By more detrimental we mean that we would prefer to retain stutter speech episodes if eliminating them means losing other lexical content. Therefore, we will explore the options available to us which help with the remediation process without cognitive meaning compromise. The level of tolerance to collateral lexical damage incurred from our remediation approach and attempts to improve a message prosodic quality must be carefully considered.

In the following remedy improvement discussion, we will use classifier prediction *confusion matrix* results and derived measurements to help frame the proposed solution qualitatively, and quantitatively.

Table 1. Confusion matrix

		Prediction outcome	
		p	n
actual value	p'	True Positive	False Negative
	n'	False Positive	True Negative

The confusion matrix Table 1 provides general measurements which apply to classification prediction models. We will describe the confusion matrix and measurements as they relate to our proposed remediation environment in general and the improvement covered in this section.

In Table 1 the prediction outcome corresponds to the segment evaluation in our classification, column *p* represents a segment evaluation of *positive*; a *positive* evaluation means that a sound episode which was extracted as a candidate for removal is evaluated as a *blocked* utterance segment and must, according to our classifier prediction, be removed from the speech.

The *n* column in the Table 1 confusion matrix represents a candidate speech episode which was presented as candidate for deletion, but the classifier prediction evaluated it to be a segment with lexical content and therefore, the classifier

indicates that it must be retained in the recording. The negative (n) column will be used to denote candidate speech episodes which were presented as candidates for deletion, but the classifier predictions evaluated them to be segments with lexical content and therefore, they should be retained in the recording. So, p means that a classifier suggests a segment must be removed and n indicates that a classifier judged a segment as *negative for removal* and thus it must remain in the recording. The *actual value* rows consist of the results obtained from the manual segment labeling effort. Manual labeling is assumed to provide accurate results, and *True* or *False* according to that assumption.

The p' row represents a speech extracted candidate segment which, when manually reviewed, was labeled as a *blocked* stutter segment and contained no lexical meaning. If we had a perfect candidate extraction system, all p' segments would be predicted as such by our classifier(s) and consequently removed.

The n' row represents speech candidates presented for removal consideration but when *manually* reviewed they were labeled as lexically meaningful and therefore should ideally be excluded from the list segments to remove from the speech.

The matrix cells are represented by four squares: *True Positive*, *False Positive*, *False Negative*, *True Negative* at the intersection of the *actuals* and *prediction*.

True Positive represents the number of candidate segments that were manually reviewed and determined to be of no value to the speech and evaluated by the trained classifier model as “bad”, lexically void, segments. This is a case where both the *actual/manual* accurate values and the classifier *predicted* values agree.

True Negative represents the number of speech segments manually labeled as lexically meaningful hence they should be retained; and, consistently, these segments were classified as lexically meaningful by the trained model. This, also, is a *prediction* outcome that agrees with the *actual* value of the segment.

The *True Positive* and *True Negative* cells represent the desired outcome and pose no concern; if all our samples fell in these two categories we would consider our classifier to be performing in an absolutely optimal way.

A *False Negative*, means that the segment is manually labeled devoid of lexical meaning (e.g. “Positive”) but the trained classifier was not able to detect that and as a result it assigned it a class of *Negative*, thus a disfluent segment, and it will be left in the recording and no repair will be performed.

The *False Positive* classifications occur when the manual, accurate, label indicates that the candidate segment is a *negative* segment implying that the segment contains semantic meaning, yet the classifier indicates that the segment should be deleted. The *False Positive* cells count represents the total number of samples that fell in that category at time of prediction. These *False* classifications cause a loss of speech semantic content.

The *False Negative* and *False Positive* cells in the confusion matrix represent predictions for which we want to insightfully fine tune our model classification capabilities, thereby improving the practical usability of our classifier.

False Positive conditions lead to speech content compromise due to removal of speech parts which contain semantic information. *False Positive* mistakes are fundamentally detrimental to the message and should be avoided if possible, albeit at a price. Avoiding a False Positive means that we will incur additional *False Negatives*; it is a trade off because the classifier is generally tuned to minimize the total number of misclassification thus maximizing accuracy. One can assume that by further tuning the classifier to minimize the total number of False Positives would result in adding proportionately more *False Negatives*.

Deciding on our level of tolerance for lexical loss is of central importance to the repair we propose. How many bad episodes are we willing to leave in a recording to avoid losing a good segment is an essential question to the work body of this research. Namely, how many false negatives would we be willing incur to avoid one *False Positive*.

To realize our tolerance for *False Positives*, we consider the False Positive Rate (*FPR*), False Negative Rate (*FNR*) and *Accuracy*. False Positive Rate (*FPR*) represents the percentage of *Negative* samples that were classified positively relative to the total number of *Negative* samples.

The lower our *FPR* is, the less mistakes our classifier would have made in classifying segments for removal when those should have been retained in the speech. Therefore, we would like the *FPR* to be as low as possible; and since we would like to compare our measures to the accuracy which ranges between 0 and 100%, we will consider Specificity (*SPC*) instead of *FPR* ($SPC = 1 - FPR$). When working with *SPC* we would aim to maximize *SPC* to a value as close to 100% as possible.

Also we will consider the False Negative Rate (*FNR*). *FNR* is the percentage of samples classified as negative when the actual class is *Positive*; these are segments that must be deleted, however our classifier has classified them for retention. These mistakes are not as damaging to the speech and will have a lesser priority. We would be willing to trade multiple *False Negatives* for one *False Positive*. For ease of comparison with *Accuracy* and *Specificity*, we will be considering the True Positive Rate ($TPR = 1 - FNR$).

There are multiple ways to perform classifications that tilt the balance in favor of more *False Negatives* than *False Positives* (or the opposite).

1. *Cost based training*: Train individual classifiers by placing a higher penalty on *False Positives* (*FP*) than *False Negatives* (*FN*) thus creating models which are less prone to *FPs*.
2. *Sampling technique*: During the classification inflate the number of positive samples in the training sets by using same Positive samples multiple times, thereby resulting in a higher count of Positive samples. This approach, also, creates classifier models with higher sensitivity to *FP* than to *FN* classifications.
3. *Probability cutoff*: During training adjust the probability cutoff for classifier models to favor minimizing *False Positives* as opposed to being optimized to minimize both *FNs* and *FPs* combined.

4. *Voting consensus*: One could also rely on a voting approach where one would take pre-trained classifiers and use a consensus among multiple classifiers to determine a final class.

Because we had many tuned and trained classifiers ready for use, we chose the fourth approach and experimented with a voting mechanism that utilizes all classifier results to consider collectively, and reach an optimal FPR. We chose to devise a voting mechanism that utilizes the result of 14 classifiers to extract a subset which, when combined, provided all we needed to perform the next steps.

In this case, we chose to vary a percentage of votes threshold as the method to decide whether a segment should be classified Positive or Negative, i.e. *Delete* or *Retain* a segment. In preparation of the dataset, we averaged the results of all classifiers and varied the decision cutoff threshold to classify every sample as Positive or Negative.

For this experimentation, we used the models that were trained and tuned within the ‘caret’ package with a balanced dataset described in prior sections. The results from the models are combined with the actual label which consists of 15 columns; one actual class (e.g. the manual label) and the 14 classifiers results.

We used the resulting table of 15 columns to evaluate the impact of changing the voting threshold on the *Specificity (SPC)*, *True Positive Rate (TPR)* and *Accuracy*. As previously stated, because false positives are especially detrimental to the remediation process, we wanted to maximize *SPC* while maintaining acceptable *Accuracy* and *TPR* levels. To make *FPs* less frequent, we intuitively expected that the higher the majority vote requirement we impose on the *votes-based* classifier, the less *FPs* (e.g. High *SPC*) we will end up with.

As shown in Table 2 we varied the voting threshold (Θ) to range from 14/14 to 1/14, where Θ denotes the number of classifiers needed to classify a segment s as *Positive*. This means that if Θ is set to 14/14 (leftmost row in Table 2), the audio segment will be classified as *Positive* if all 14 classifiers vote *Positive*; if Θ is set to 13/14 the audio segment will be classified as *Positive* when 13 or more classifiers vote *Positive* etc.

In addition to *True Positive*, *True Negative*, *False Positive*, and *False Negative* totals, we measured the *Accuracy* rate, *Specificity*, and *TPR* corresponding to each of the 14 voting threshold results. As can be shown by Table 2, our initial intuition that to minimize *FPs* we must use higher threshold was confirmed.

The analysis of Table 2 leads to the observation that there is a reverse correlation between *FP* and *FN*. A close review of *Accuracy (Accu)*, *SPC* and *TPR* shows the best *Accuracy* results (97.81%) to occur at $\Theta = 12/14$ but the nature of our speech remediation prefers lower *FP* classes even if there is a relatively larger increase in *FN* classifications. By considering $\Theta = 13/14$ we lower the *FP* values from 19 to 6, namely, we end up with 13 less instances where we remove an episode which contains actual meaning; while increasing the *FN* classifications from 23 to 43 thereby increasing the number of stutter *blocks* that remain in the speech by 20. So, we choose to trade 13 *FPs* for 20 *FNs* and left 20 *blocks* in the recording.

Table 2. TP, TN, FP, FN with specificity, TPR, accuracy

						1-FPR	1-FNR
Vote Θ	TP	TN	FP	FN	Accu	SPC	TPR
14/14	1601	198	1	121	93.65%	99.50%	92.97%
13/14	1679	193	6	43	97.45%	96.98%	97.50%
12/14	1699	180	19	23	97.81%	90.45%	98.66%
11/14	1712	157	42	10	97.29%	78.89%	99.42%
10/14	1715	132	67	7	96.15%	66.33%	99.59%
9/14	1717	99	100	5	94.53%	49.75%	99.71%
8/14	1720	84	115	2	93.91%	42.21%	99.88%
7/14	1720	63	136	2	92.82%	31.66%	99.88%
6/14	1721	50	149	1	92.19%	25.13%	99.94%
5/14	1722	40	159	0	91.72%	20.10%	100.00%
4/14	1722	33	166	0	91.36%	16.58%	100.00%
3/14	1722	23	176	0	90.84%	11.56%	100.00%
2/14	1722	0	199	0	89.64%	0.00%	100.00%
1/14	1722	0	199	0	89.64%	0.00%	100.00%

Such a trade off is reasonable when considering the high negative impact False Positives have on a recording and we find that varying a voting cut off threshold brings considerable improvement to the overall practicality of our system.

The effect of moving the Θ threshold is further illustrated in Fig. 3 where the line graph depicted over a narrow range of Θ (14/14 to 7/14) and a narrow range of *Specificity*, *TPR* and *Accuracy* (70% to 100%) visually magnifies the impact of Θ and helps us confirm the soundness of our decision to use a Θ value of 13/14.

3 Experiments and Results

Our approach proved to be effective in eliminating the vast majority of voice blocks when applied to stuttered speech. The results of disfluent speech remediation consisted of the elimination of anomalous speech and a reduction in speech length between 60% for the most severe stuttered speech to about 25% for mildly stuttered speeches.

The effectiveness of our remediation process is highly dependent on the threshold of the speech intensity tolerance during the potentially undesired segments identification. As we have stated earlier, we used two different thresholds for the sound intensity for the initial phase - not to be confused with the voting threshold. On the one hand, the lower threshold (90% of average) resulted in a considerably less undesired candidates as shown in Table 3; although most of the undesired candidates extracted using the lower threshold were actually

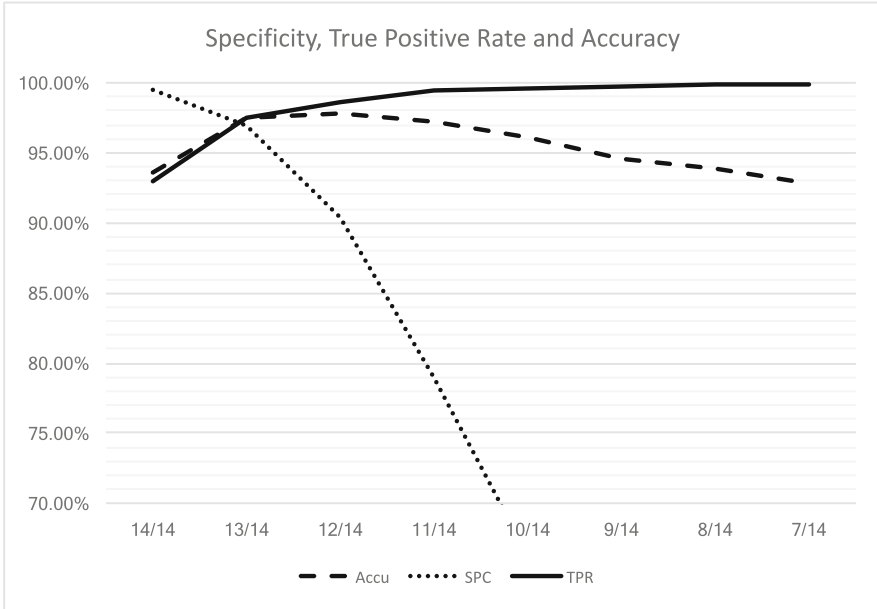


Fig. 3. Specificity, TPR, Accuracy (Accu) according the voting threshold

Table 3. Segment count

	90% of speech avg sound volume threshold	95% of speech avg sound volume threshold
0.6 s	195	211
0.8 s	149	159
1.0 s	114	130
1.2 s	83	93

true undesired, there were many instances where stutter segments were not detected (due to the low threshold). On the other hand, the higher threshold (95%) resulted in a much higher number of potentially undesired segments, some of which were false undesired; however, most of the stuttered segments were detected in the first phase. Therefore, the value of the second phase (training and testing phase), which identifies true undesired and false undesired from potentially undesired segments, is most useful when the threshold is high, and when most of the actual stutter segments are detected, even if that means potentially marking some segments as potentially undesired while in reality they are false undesired, during the first phase.

We have subsequently added a component to this system of remediation which helped us to enhance the usability of the system. Because a good remediation process should lose little to no semantic meaning we implemented a *False*

Positives restricting subsystem which relies on a voting cutoff approach. This enhancement would only tag a segment for removal if the vote for removal is almost unanimous amongst all qualifiers. Although our accuracy was negatively affected by this component, the results of this experimentation proved to be effective in reducing *False Positives*.

The second threshold value that we used during the extraction of the potentially undesired sound segments was the minimum time duration, starting with 0.6s and ending with 1.2s. It is worth noting that we tried shorter duration time frames with little success, causing the comprehensibility of the speech to be diminished; we found 0.6s minimum silence duration to be the lowest value that can be used in our experiments.

We used two different approaches for treating our data with each classifier. The first approach is to build one classifier for our data after balancing the labels regardless of the minimum duration and intensity threshold values; in other words, we combine all the segments extracted from all eight different unique pairs of minimum duration and intensity threshold, and build our classifier accordingly. The balancing approach we chose to apply for the balancing of the data consisted of randomly excluding data tuples of the over-represented class G.

The second approach is to build a single classifier for each unique pair, then average the accuracy and confusion matrix results. Table 4 shows the results obtained using the first approach, while Table 5 shows the results obtained from using the second approach for the two classifiers that performed the best using separate classifiers for every unique pair; Random Forests and C5.0.

Using 10-fold cross validation, we trained six different classifiers: C5.0, Neural Networks, Recursive Partitioning, Random Forests, Polynomial SVM, and AdaBoost. During the training of our classifiers, we used the ‘caret’*R* package because of its built-in tuning functionality. The parameters to build the respective models were chosen based on highest accuracy. We have listed the ‘caret’ package chosen parameters below:

Neural Network (nnet): Tuned model parameters when training on the balanced dataset: size = 5 and decay = 0.1. Tuned model parameters when training on the entire dataset: size = 1 and decay = 0.1. Size represents the number of

Table 4. Results obtained from building a single classifier using all of segments

	True positive rate	True negative rate	Accuracy
C5.0	100%	94%	97.4%
Neural networks	75.6%	55.9%	67.7%
Recursive partitioning	92.6%	94.2%	93.2%
Random forests	98.4%	91.7%	95.8%
Polynomial SVM	89.9%	75.6%	84.4%
AdaBoost	98.3%	93.8%	96.4%

Table 5. Results obtained from using a single classifier for each pair of minimum duration and intensity threshold, and averaging the results

	Random forest	C5.0 Classifier
True positive rate (Averaged)	95.3 %	94.8%
True negative rate (Averaged)	82.3 %	76.6%
Classifier with highest true positive rate	1.2 minimum duration, 90% intensity threshold	0.8 minimum duration, 90% intensity threshold
Classifier with highest true negative rate	1.2 minimum duration, 90% intensity threshold	1.2 minimum duration, 90% intensity threshold
Classifier with lowest true positive rate	1.2 minimum duration, 95% intensity threshold	1.0 minimum duration, 95% intensity threshold
Classifier with lowest true negative rate	1.0 minimum duration, 95% intensity threshold	1.0 minimum duration, 95% intensity threshold

units in the hidden layer and can be zero if there are skip-layer units. The decay parameter represents weight decay.

Random Forest (rf): Tuned model parameters when training on the balanced dataset: $mtry = 2$. Tuned model parameters when training on the entire dataset: $mtry = 56$. The parameter $mtry$ represents the number of variables randomly sampled as candidates at each split.

SVM Polynomial (svmPoly): Tuned model parameters when training on the balanced dataset: $degree = 3$, $scale = 0.01$ and $C = 1$. Tuned model parameters when training on the entire dataset: $degree = 3$, $scale = 0.01$ and $C = 1$. The degree parameter represents the polynomial degree of the kernel function. The scale is the scaling parameter of the polynomial and tangent kernel. C is the cost regularization parameter.

Adaptive Boosting (AdaBag): Tuned model parameters when training on the balanced dataset: $mfinal = 100$ and $maxdepth = 3$. Tuned model parameters when training on the entire dataset: $mfinal = 150$ and $maxdepth = 3$. The $mfinal$ parameters represents the number of iterations for which boosting is run or the number of trees to use. $Maxdepth$ is the maximum depth of any node of the final tree.

Recursive Partitioning (rpart): Tuned model parameters when training on the balanced dataset: $cp = 0.03797468$. Tuned model parameters when training on the entire dataset: $cp = 0.05970149$. Cp is the complexity parameter; any split that does not decrease the overall lack of fit by a factor of cp is not attempted. The main role of this parameter is to save computing time by pruning off splits that are evidently not worthwhile.

As can be seen in Table 4, when we balance our data we reduce the number of samples in the dataset, this big reduction in the number of tuples seems to have a detrimental impact on our neural network classifier model, since neural network training is best accomplished with large datasets.

4 Conclusion and Future Work

In summary, we described a system which can be successfully used to reduce stuttered speech disfluency by removing certain potentially undesired speech segments. The dataset consisted of speech and speaker metadata and speech signal statistics. The remedied speeches showed a marked improvement over the original speeches.

The innovation in our system is two-fold. First, both the scope of data collection and the classification is predetermined and designed with the singular objective of determining whether a possible action must be performed or not; in our case, we only collected potentially undesired candidate segments that might be removed, and extracted features which are then used to classify the segments with the sole purpose of deciding whether a segment needs to be removed or not. Secondly, the data collected represents different examination perspectives of single instances.

As a practical enhancement to our system we implemented a voting based classification system to reduce the possibility of speech meaning loss by tilting the model in favor of less meaning loss at the expense of missed removal of blocked segments. This also was interesting in outcome and continued such explorations would likely bring practical usability enhancements to the system.

A specific use of a system such as the one presented in this work is to remedy a disfluent speech with blocks for a speaker who wishes to make his (or her) speech easy to listen to and better understood. Remediation makes a speech easier to listen to with minimal inconvenience to the listener and minimal need for re-recording.

The concept of single action based intelligent solutions can help systems utilize compartmentalized machine learning and classification solutions. Such scope-specific machine-learned actions can provide lightweight and fast independent action prediction tool that can be chained together for more complex tasks. The process of utilizing the use of multiple perspectives can be employed to optimize and devise an adaptive approach to data gathering, where the data collection method and scope are optimized according to the metadata and condition of the subject on hand.

In future work, we recommend utilizing various extraction thresholds to fine tune the feature collection to the speech condition, which would better streamline the remediation process. Another extension to disfluent speech remediation would be to eliminate speech interjections by identifying the voiced and unvoiced segments of speech and then singling out those voiced sound segments with interjection episode characteristics for removal.

References

1. Ai, O.C., Hariharan, M., Yaacob, S., Chee, L.S.: Classification of speech dysfluencies with MFCC and LPCC features. *Expert Syst. Appl.* **39**(2), 2157–2165 (2012)
2. Boersma, P.: Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In: *Proceedings of the Institute of Phonetic Sciences*, vol. 17, no. 1193 (1993)
3. Chee, L.S., Ai, O.C., Yaacob, S.: Overview of automatic stuttering recognition system. In: *Proceedings of the International Conference on Man-Machine Systems*, Batu Ferringhi, Penang Malaysia, pp. 1–6, October 2009
4. Fook, C.Y., Muthusamy, H., Chee, L.S., Yaacob, S.B., Adom, A.H.B.: Comparison of speech parameterization techniques for the classification of speech disfluencies. *Turkish J. Electr. Eng. Comput. Sci.* **21**, no. Sup. 1 (2013)
5. Hariharan, M., Chee, L.S., Ai, O.C., Yaacob, S.: Classification of speech dysfluencies using LPC based parameterization techniques. *J. Med. Syst.* **36**(3), 1821–1830 (2012)
6. Honal, M., Schultz, T.: Automatic disfluency removal on recognized spontaneous speech-rapid adaptation to speaker dependent disfluencies. In: *ICASSP*, vol. 1, pp. 969–972 (2005)
7. Honal, M., Schultz, T.: Correction of disfluencies in spontaneous speech using a noisy-channel approach. In: *Interspeech* (2003)
8. Howell, P., Davis, S., Bartrip, J.: The UCLASS archive of stuttered speech. *J. Speech Lang. Hear. Res.* **52**(2), 556–569 (2009)
9. Km, R.K., Ganesan, S.: Comparison of multidimensional MFCC feature vectors for objective assessment of stuttered disfluencies. *Int. J. Adv. Netw. Appl.* **2**(05), 854–860 (2011)
10. Lease, M., Johnson, M., Charniak, E.: Recognizing disfluencies in conversational speech. *IEEE Trans. Audio Speech Lang. Process.* **14**(5), 1566–1573 (2006)
11. Liu, Y., Shriberg, E., Stolcke, A., Harper, M.P.: Comparing HMM, maximum entropy, and conditional random fields for disfluency detection. In: *Interspeech*, pp. 3313–3316 (2005)
12. Ravikumar, K.M., Rajagopal, R., Nagaraj, H.C.: An approach for objective assessment of stuttered speech using MFCC. In: *The International Congress for Global Science and Technology*, p. 19 (2009)
13. Świetlicka, I., Kuniszyk-Józkowiak, W., Smółka, E.: Hierarchical ANN system for stuttering identification. *Comput. Speech Lang.* **27**(1), 228–242 (2013)
14. Raghavendra, M., Rajeswari, P.: Determination of disfluencies associated in stuttered speech using MFCC feature extraction. *Comput. Speech Lang, IJEDR* **4**(2), 2321–9939 (2016)
15. Czyzewski, A., Kaczmarek, A., Kostek, B.: Intelligent processing of stuttered speech. *J. Intell. Inf. Syst.* **21**, 143–171 (2003)



Identifying lncRNA-Disease Relationships via Heterogeneous Clustering

Emanuele Pio Barracchia¹ , Gianvito Pio¹  , Donato Malerba^{1,2} ,
and Michelangelo Ceci^{1,2} 

¹ Department of Computer Science, University of Bari Aldo Moro,
Via Orabona, 4, 70125 Bari, Italy
e.barracchia@studenti.uniba.it,

{gianvito.pio,donato.malerba,michelangelo.ceci}@uniba.it

² CINI - Consorzio Interuniversitario Nazionale per l'Informatica, Bari, Italy

Abstract. High-throughput sequencing technology led significant advances in functional genomics, giving the opportunity to pay particular attention to the role of specific biological entities. Recently, researchers focused on long non-coding RNAs (lncRNAs), i.e. transcripts that are longer than 200 nucleotides which are not transcribed into proteins. The main motivation comes from their influence on the development of human diseases. However, known relationships between lncRNAs and diseases are still poor and their in-lab validation is still expensive. In this paper, we propose a computational approach, based on heterogeneous clustering, which is able to predict possibly unknown lncRNA-disease relationships by analyzing complex heterogeneous networks consisting of several interacting biological entities of different types. The proposed method exploits overlapping and hierarchically organized heterogeneous clusters, which are able to catch multiple roles of lncRNAs and diseases at different levels of granularity. Our experimental evaluation, performed on a heterogeneous network consisting of microRNAs, lncRNAs, diseases, genes and their known relationships, shows that the proposed method is able to obtain better results with respect to existing methods.

1 Introduction

High-throughput sequencing technology, alongside new computational methods, has been crucial for rapid advances in functional genomics. Among the most important results achieved by exploiting these new technologies, there is the discovery of thousands of non-coding RNAs (ncRNAs). Since their function appears to be pivotal for the fine-tuning of the expression of many genes [3], in the last decade, the number of papers reporting evidences about ncRNAs involvement in human complex diseases, such as cancer, has grown at an exponential rate. Among the different classes of ncRNAs, the most investigated one is that of microRNAs (miRNAs), which are small molecules that regulate the expression of genes through the modulation of the translation of their transcripts [7]. Much less is known about the functional involvement of long non-coding RNAs

(lncRNAs), i.e. non-coding transcripts which are longer than 200 nucleotides, that have been recently discovered to have a plethora of regulatory functions [11]. However, the number of lncRNAs for which the functions are known is still quite poor and their in-lab validation requires large resources. Thus, assessing the role and, especially, the molecular mechanisms underlying the involvement of lncRNAs in human diseases, is not a trivial task.

In the last few years, there were some attempts to computationally predict the relationships among biological entities, such as genes, miRNAs, lncRNAs, diseases, tissues, etc. An example can be found in [14], where the authors propose an approach to learn to combine the outputs of several algorithms for the prediction of miRNA-gene interactions. A more sophisticated approach has been proposed in [4], where the authors adopt the multi-view learning framework for the reconstruction of gene-gene interaction networks.

Focusing on the identification of relationships involving diseases, in [16] the authors propose a method to identify possible relationships between lncRNAs and diseases, by exploiting a bipartite network and a propagation algorithm. Analogously, in [1] the authors propose the method *ncPred* which exploits a tripartite graph representing known ncRNA-gene and gene-disease associations. Such a graph is analyzed by adopting a multi-level resource transfer technique that, at each step, takes into account the resource transferred in the previous one. For each detected interaction, the algorithm associates a score indicating its degree of certainty. Both these methods, however, cannot exploit additional information associated with the involved biological entities as well as other entities that are related to the considered ones (e.g., genes, miRNAs, tissues, etc.).

In this paper, we present a novel method for the identification of previously unknown relationships between diseases and lncRNAs, which extends the heterogeneous clustering approach we proposed in [15]. In particular, the proposed method is able to identify heterogeneous clusters from heterogeneous networks, where nodes are biological entities (each associated with their own features) and edges represent known relationships among them (see Fig. 1). Then, the identified clusters are exploited to predict the possible existence of unknown relationships between lncRNAs and diseases falling in the same clusters. This approach is motivated by the fact that lncRNAs and diseases will fall in the same clusters if they appear similar according to their features and their relationships with the other analyzed entities. Therefore, the main advantage of the approach proposed in this paper comes from its ability to globally take into account the complex network of interactions involving different biological entities. Moreover, the proposed algorithm has the advantage of identifying possibly overlapping and hierarchically organized clusters, since (i) the same lncRNA/disease can be involved in multiple networks of relationships and (ii) as shown in [12], clusters at different levels of the hierarchy can describe more specific or more general relationships and cooperation activities. In the following section, we briefly describe our clustering method and its exploitation to identify unknown lncRNA-disease relationships, while in Sect. 3 we report the results of our experiments. Finally, in Sect. 4, we draw some conclusions and outline the ongoing work.

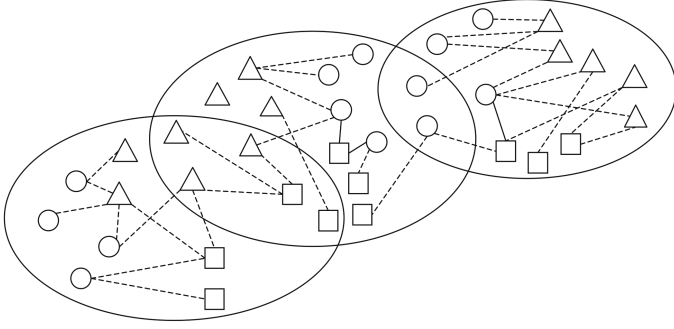


Fig. 1. An example of a heterogeneous network, where different shapes represent different node types. Circles represent possible heterogeneous clusters.

2 Method

In the following, we introduce the notation and some useful definitions.

Definition 1 (Heterogeneous network). A heterogeneous network is a network $G = (V, E)$, where V is the set of nodes and E is the set of edges among nodes, where both nodes and edges can be of different types. Moreover:

- each node $v' \in V$ is associated to a single node type $t_v(v') \in \mathcal{T}$, where \mathcal{T} is the finite set $\{T_p\}$ of all the possible types of nodes in the network;
- each node type T_p implicitly defines a subset of nodes $V_p \subseteq V$;
- a node type T_p defines a set of attributes $\mathcal{X}_p = \{X_{p,1}, X_{p,2}, \dots, X_{p,m_p}\}$;
- an edge e between two nodes v' and v'' is associated to an edge type $R_j \in \mathcal{R}$, where \mathcal{R} is the finite set $\{R_j\}$ of all the possible edge types in the network. Formally, $e = \langle R_j, \langle v', v'' \rangle \rangle \in E$, where $R_j = t_e(e) \in \mathcal{R}$ is its edge type;
- an edge type R_j defines a subset of edges $E_j \subseteq (V_p \times V_q) \subseteq E$;
- node types \mathcal{T} are partitioned into \mathcal{T}_t (target), i.e. considered as target of the clustering/prediction task, and \mathcal{T}_{tr} (task-relevant). Only nodes of target types are actually clustered and considered in the identification of new relationships, on the basis of all the nodes.

Definition 2 (Heterogeneous cluster). We define a heterogeneous cluster, or multi-type cluster, as $G' = (V', E')$, where: $V' \subseteq V$; $\forall v' \in V', t_v(v') \in \mathcal{T}_t$ (nodes in the clusters are only of target types); $E' \subseteq (E \cup \hat{E})$ is a set of edges (among the nodes in V') belonging either to E or to a set of edges \hat{E} containing *extracted* edges, which relate nodes that are not directly connected in the original network.

Definition 3 (Hierarchical organization of clusters). A hierarchy of heterogeneous clusters is defined as a list of hierarchy levels $\{L_1, L_2, \dots, L_k\}$, each of which consisting of a set of heterogeneous and possibly overlapping clusters.

In this specific application domain, target nodes are those representing lncRNAs and diseases. Therefore, we distinguish two distinct sets of nodes T_l and T_d , representing the set of lncRNAs and the set of diseases, respectively. Our task then consists in the identification of a hierarchy of clusters $\{L_1, L_2, \dots, L_k\}$ and of a function $\psi^{(w)} : T_l \times T_d \rightarrow [0, 1]$ for each hierarchy level L_w , which, for each lncRNA-disease pair, returns a score indicating its degree of certainty.

In the following, we describe our solution consisting of three steps: (i) identification of the strength of relationships among nodes in T_l and T_d , which will define the set of extracted edges \hat{E} ; (ii) construction of a hierarchy of (possibly overlapping) heterogeneous clusters; (iii) identification of the functions $\psi^{(w)}$ for the prediction of previously unknown relationships.

2.1 Identification of the Strength of the Relationship Among Nodes

We first estimate the strength of the relationship of all the possible lncRNA-disease pairs, following the idea we proposed in [15]: for each pair (l_i, d_j) , we compute the score $s(l_i, d_j)$ by analyzing the indirect relationships in which the lncRNA l_i and the disease d_j are involved. In particular, as in [15], we adopt the concept of *meta-path*, i.e., the set of sequences of nodes which follow the same sequence of edge types. For each meta-path P between l_i and d_j , we compute a score $pathscore(P, l_i, d_j)$ representing the strength between l_i and d_j following the meta-path P . Since several meta-paths can be identified between two objects in the network, possibly with unlimited length (in presence of cycles), we have to identify a strategy to assign a single score to each lncRNA-disease pair. The strategy we considered is inspired by the classical formulation of fuzzy sets [17]. In particular, since $s(l_i, d_j)$ should measure the degree of certainty of the relationship between l_i and d_j , we consider the scores computed over each meta-path P (i.e., $pathscore(P, l_i, d_j)$) as the degree of certainty estimated according to P . Since the relationship between l_i and d_j can be considered certain if there exists at least one meta-path which proves its certainty (or, in other words, the certainty of the relationship corresponds to the highest certainty showed over the meta-paths), we compute $s(l_i, d_j)$ as follows:

$$s(l_i, d_j) = \max_{P \in \text{metapaths}(l_i, d_j)} pathscore(P, l_i, d_j) \quad (1)$$

where $\text{metapaths}(l_i, d_j)$ is the set of the c shortest paths connecting l_i and d_j , and $pathscore(P, l_i, d_j)$ is the degree of certainty of the relationship between l_i and d_j according to the meta-path P .

In order to compute $pathscore(P, l_i, d_j)$, we represent each meta-path P as a finite set of sequences of nodes. If a sequence in P connects l_i and d_j , then $pathscore(P, l_i, d_j) = 1$. Otherwise, following the same strategy introduced before, it is computed as the maximum similarity between the sequences which start with l_i and the sequences which end with d_j (see Fig. 2).

The similarity between two sequences seq' and seq'' is computed according to the attributes of all the nodes involved in the two sequences. Following [6], the similarity between two values of an attribute x , i.e., $s_x(seq', seq'')$,

#Seq	disease_id	disease_att1	disease_att2	lncRNA_id	lncRNA_att1	lncRNA_att2
1	d ₁	0.5	a	l ₂	0.9	x
2	d ₄	0.5	c	l ₃	0.5	y
3	d ₃	0.6	c	l ₂	0.9	x
4	d ₃	0.6	c	l ₁	0.3	x
5	d ₃	0.6	c	l ₈	0.8	z
6	d ₂	0.3	b	l ₃	0.5	y
7	d ₅	0.1	b	l ₁	0.7	y

Fig. 2. An example of analysis of the sequences associated to the lncRNA l_3 and to the disease d_3 . In the example, sequences 2 and 6 (in yellow) are associated to the lncRNA l_3 , and sequences 3, 4 and 5 (in green) are associated to the disease d_3 . The algorithm pair-wisely compares the two sets of sequences (sequences in yellow and sequences in green) and computes the degree of certainty between l_3 and d_3 as the maximum similarity between two sequences. (Color figure online)

is computed as follows: If x is a numerical attribute, then $s_x(seq', seq'') = 1 - \frac{|val_x(seq') - val_x(seq'')|}{max_x - min_x}$ (min_x and max_x are the minimum and maximum values, respectively, observed for the attribute x); when x is not numeric, then $s_x(seq', seq'') = 1$ if $val_x(seq') = val_x(seq'')$, 0 otherwise.

It is noteworthy that some node types may not be involved in any meta-path. In order to exploit the information conveyed by these nodes, we add an aggregation of their attribute values to the nodes that are connected to them and that appear in at least one meta-path. Such an aggregation considers values coming from directly or indirectly (up to a predefined *depth* of analysis) connected nodes. For this purpose different aggregation functions could be used. Following [15], we use the *arithmetic mean* for numerical attributes, the *mode* for non-numerical attributes and limit the depth of analysis for the aggregation to 2.

2.2 Construction of the Hierarchy of Heterogeneous Clusters

Once all the possible pairs are identified, each associated with its degree of certainty, we first build a set of (possibly overlapping) clusters in the form of bicliques to be used in the subsequent step. A cluster is in the form of a biclique if all the lncRNA-disease pairs in the cluster have a score above a given threshold $\beta \in [0, 1]$. The algorithm consists of the following steps:

- (i) A filtering phase which keeps only the pairs with a score greater than (or equal to) β . The result is the subset of pairs $\{(l_i, d_j) | s(l_i, d_j) \geq \beta\}$.
- (ii) An initialization step which identifies the initial set of bicliques, each consisting of a lncRNA-disease pair in $\{(l_i, d_j) | s(l_i, d_j) \geq \beta\}$.
- (iii) A process that iteratively merges two clusters G' and G'' into a new cluster G''' . The initial set of clusters is regarded as a list and is sorted according to an ordering relation $<_c$ that reflects the quality of the clusters. Each cluster G' is merged with the first cluster G'' in the list leading to a merged cluster

G''' which still is a biclique. This step is repeated until no more merging can be performed. The obtained result is the first hierarchy level L_1 .

The ordering relation $<_c$ is based on the cluster *cohesiveness*, defined as:

$$h(G) = \frac{1}{|pairs(G)|} \cdot \sum_{(l_i, d_j) \in pairs(G)} s(l_i, d_j) \quad (2)$$

where $pairs(G)$ is the set of all the possible lncRNA-disease pairs (both known and unknown) in the cluster.

This measure actually corresponds to the average score of the relationships in the cluster. Since, in our case, the score represents a degree of certainty, the cluster cohesiveness can be considered as an indicator of the degree of certainty of the global interactions among the group of lncRNAs and diseases in the cluster. Therefore, we formally define the ordering relation $<_c$ as follows:

$$G' <_c G'' \iff h(G') > h(G'') \quad (3)$$

Once the first level L_1 of the hierarchy has been identified, the other levels are built by evaluating whether some pairs of clusters (bicliques, in L_1) can be reasonably merged. The approach is similar to that used to obtain the first level of the hierarchy. The main difference is that, instead of working on bicliques, we work on generic clusters, where the score associated to each pair is not necessarily greater than β . Due to this difference, we use a different criterion for the identification of candidates for merging which is inspired by the research in hierarchical co-clustering. In this research, one of the commonly used stopping criterion is based on a predefined threshold applied to the quality constraint that must be satisfied in order to merge two clusters [12]. Analogously, in our approach, two clusters G' and G'' are merged into a cluster G''' if $h(G''') > \alpha$, where α is a user defined threshold on the cluster cohesiveness. Note that low values of α lead to a higher number of mergings and, accordingly, to less clusters containing a higher number of objects.

We repeat the process until no merging is possible and return the obtained hierarchy of heterogeneous clusters $\{L_1, L_2, \dots, L_k\}$, according to Definition 3.

2.3 Prediction of Unknown Relationships

After building the hierarchy of clusters, we identify possibly unknown relationships for each hierarchical level. In particular, the prediction is performed by assigning each possible lncRNA-disease pair with a degree of certainty computed as the cohesiveness of the cluster in which it falls. More formally, given $C_{ij}^{(w)}$ the cluster in which the lncRNA l_i and the disease d_j fall in the w -th hierarchical level, we compute the final degree of certainty of the relationship as:

$$\psi^{(w)}(l_i, d_j) = h\left(C_{ij}^{(w)}\right). \quad (4)$$

When the lncRNA l_i and the disease d_j appear in multiple clusters, i.e., $C_{ij}^{(w)}$ is a list of clusters, we combine their cohesiveness to obtain the final degree of certainty. Baseline combination strategies can be the maximum, the minimum and the average. In this work, we propose to adopt a different combination function, which rewards those cases in which the pair appears in several highly cohesive clusters (indicating a higher degree of certainty). In details, inspired by the evidence combination (EC) strategy proposed in [10], given $C_{ij}^{(w)} = [C_1, C_2, \dots, C_m]$, the list of the clusters in which the lncRNA l_i and the disease d_j fall in the w -th hierarchical level, we compute $\psi^{(w)}(l_i, d_j) = ec(C_m)$, where $ec(C_m)$ is recursively defined as:

$$ec(C_m) = \begin{cases} h(C_1) & \text{if } C_m = C_1 \\ ec(C_{m-1}) + [1 - ec(C_{m-1})] \cdot h(C_m) & \text{otherwise} \end{cases} \quad (5)$$

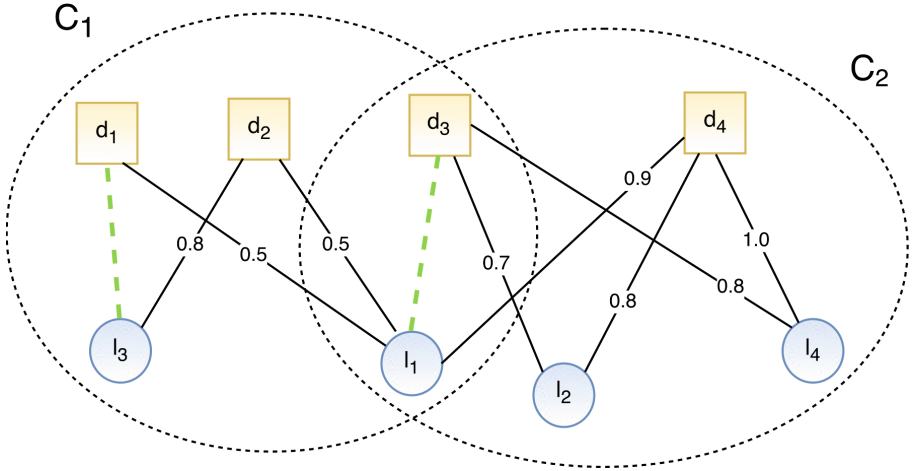


Fig. 3. Two possible clusters identified at a given hierarchical level w . Circles represent lncRNAs, while squares represent diseases. The clusters suggest two new possible relationships between l_3 and d_1 and between l_1 and d_3 .

In Fig. 3, we show an example of the prediction step, where the two clusters C_1 and C_2 , identified at the w -th hierarchical level, suggest two potential new relationships, i.e., between l_3 and d_1 and between l_1 and d_3 . The former falls only in the cluster C_1 , therefore it will be associated with a degree of certainty computed according to the cohesiveness of C_1 . Formally:

$$\psi^{(w)}(l_3, d_1) = h(C_1) = \frac{1}{3 \cdot 2} (0.8 + 0.5 + 0.5) = 0.3.$$

The latter falls in both C_1 and C_2 and its degree of certainty will be computed according to the cohesiveness of both clusters. In particular, given $h(C_1) = 0.3$

and $h(C_2) = \frac{1}{3 \cdot 2}(0.7 + 0.8 + 0.9 + 0.8 + 1.0) = 0.7$, by adopting the EC strategy (see Eq. 5), the degree of certainty of the relationship between l_1 and d_3 will be computed as:

$$\psi^{(w)}(l_1, d_3) = h(C_1) + [1 - h(C_1)] \cdot h(C_2) = 0.3 + (1 - 0.3) \cdot 0.7 = 0.79$$

3 Experiments

The proposed method has been implemented in the system LP-HCLUS (Link Prediction through Heterogeneous CLUstering). We performed our experimental evaluation in order to evaluate the effectiveness of the proposed approach on a complex biological dataset containing data about lncRNAs, miRNAs, genes and diseases, as well as their known interactions and relationships. Such a dataset, whose schema is depicted in Fig. 4, has been built by integrating several existing biological datasets:

- lncRNA-disease relationships and lncRNA-gene interactions from [5];
- miRNA-lncRNA interactions from [8];
- disease-gene relationships from DisGeNET [2];
- miRNA-gene and miRNA-disease relationships from miR2Disease [9].

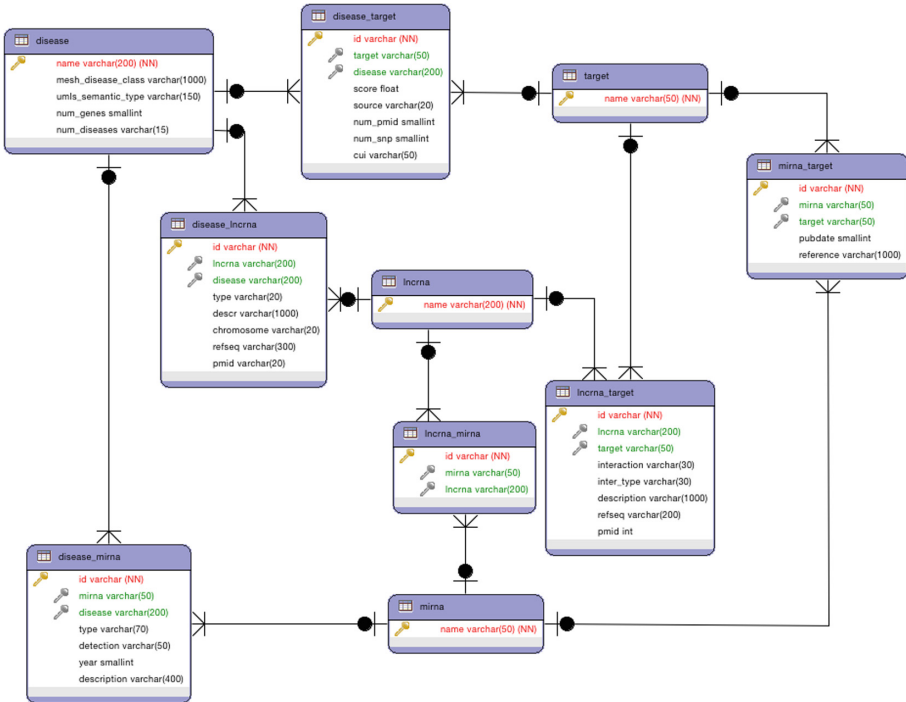


Fig. 4. UML representation of the heterogeneous network used in the experiments.

The integrated dataset consists of 7050 diseases, 507 lncRNAs, 508 miRNAs, 94527 genes, 953 interactions between diseases and lncRNAs, 2877 interactions between diseases and miRNAs, 26522 interactions between diseases and genes, 70 interactions between lncRNAs and miRNAs, 252 interactions between lncRNAs and genes, and 803 interactions between miRNAs and genes.

We adopted the 10-fold cross validation on the set of known lncRNA-disease relationships. Due to the absence of negative examples, following the approach adopted in [13], we averaged the results obtained in terms of $recall@K$, i.e., the recall measured by considering only the top- k returned relationships. In detail, we produce a ranking of the predicted interactions by sorting them in descending order with respect to their degree of certainty and compute $recall@K = \frac{TP_k}{TP_k + FN_k}$, where TP_k (respectively, FN_k) is the number of validated lncRNA-disease relationships that were (respectively, were not) predicted in the first K returned interactions. Since the most appropriate value of K cannot be known in advance, we plot the obtained $recall@K$ by varying the value of K .

LP-HCLUS has been run by considering 3 shortest paths for each lncRNA-disease pair (i.e., $c = 3$). We collected the results obtained with the maximum (MAX), the minimum (MIN), the average (AVG) and the evidence combination (EC) strategies to combine the degree of certainty of relationships identified in multiple clusters, focusing on the first 3 levels of the identified hierarchies which, according to [12], lead to the best results.

As competitor systems, we considered the following approaches:

- A variant of the biclustering algorithm **HOCCLUS2** [12], which is able to solve the link prediction task. HOCCLUS2 has similar characteristics with respect to the clustering approach proposed in this paper, i.e., it is able to extract a hierarchy of (possibly overlapping) clusters. However, it does not allow to take into account several types of objects, linked by several types of edges. Moreover, the algorithm adopted for the construction of the hierarchy of clusters is different and guided only by the cohesiveness.
- The link prediction algorithm **ncPred** [1], which is tailored for the prediction of ncRNA-disease associations.
- A baseline approach, which consists in the estimation of the degree of certainty by means of the strategy described in Sect. 2.1, i.e., without the clustering and the prediction steps. The comparison of the results with respect to this baseline approach allows us to evaluate the real contribution of the exploitation of clusters for link prediction. We call this baseline approach **LP-HCLUS w/o LP** (i.e., LP-HCLUS without Link Prediction).

We fed all the competitor methods with the set of lncRNA-disease scores computed by LP-HCLUS, since, in their original form, they are not able to analyze a complex heterogeneous network.

Since both HOCCLUS2 and LP-HCLUS require the input parameters α and β , we performed some preliminary experiments to evaluate their effect on the results. In particular, we evaluated the results with the following configurations:

$\alpha = 0.1$ and $\beta = 0.3$; $\alpha = 0.1$ and $\beta = 0.4$; $\alpha = 0.2$ and $\beta = 0.3$; $\alpha = 0.2$ and $\beta = 0.4$. By observing the results reported in Fig. 5, obtained with the EC strategy on the first three hierarchical levels, we can conclude that the results do not appear to be significantly affected by these parameters. A similar behavior was observed for the other combination strategies and for HOCCLUS2. However, since the obtained *recall@K* results appeared higher in the case of $\alpha = 0.2$ and $\beta = 0.4$, the other experiments were conducted with such values.

In Table 1, we report the results in terms of *recall@K* obtained by the considered approaches, with $K \in \{500, 1000, \dots, 5000\}$. The first conclusion that can be drawn regards the superiority of LP-HCLUS with respect to the considered competitor approaches, with all the values of K . This conclusion is even more evident for small values of K , i.e., in the first part of the ranked interactions. Moreover, by comparing the results obtained by LP-HCLUS with the baseline approach (LP-HCLUS w/o LP) we can observe a significant improvement when the clustering and the link prediction phases are adopted. This means that the strategy proposed in this paper, i.e., the identification of heterogeneous clusters and their exploitation for link prediction purposes, appears to be effective. Moreover, although the adopted variant of HOCCLUS2 is based on the same principle, it still leads to a lower *recall@K* results, emphasizing that the clustering algorithm and the adopted combination strategies proposed in this paper perform better. A further observation comes from the comparison of the results obtained by LP-HCLUS with different combination strategies. Indeed, by observing Table 1, we can conclude that the strategy based on evidence combination (EC) generally leads to the best results, especially for high values of K . This is mainly due to the fact that it rewards the interactions falling in multiple highly-cohesive clusters. This means that, on overall, predicted interactions have a higher degree of certainty (also higher than the those predicted with the strategy based on MAX), leading to a higher recall with high values of K .

A more global overview is provided in Fig. 6, where we plot the *recall@K* results of all the considered approaches, at different levels of the hierarchy. This figure shows the overall superiority of LP-HCLUS, when the strategy based on evidence combination is adopted. Moreover, it also shows that the competitors (i.e., ncPred and HOCCLUS2) and the baseline method cannot reach the recall values obtained by LP-HCLUS (very close to 1.0) even with $K = 70,000$.

A final consideration comes from the analysis of the results at different levels of the hierarchy. At this respect, we could not find a general trend in the results, in terms of *Recall@K*. However, such a measure is only able to evaluate the results quantitatively, and a deeper analysis could be necessary in order to emphasize possible differences from a qualitative viewpoint. Therefore, since we still believe that the hierarchy can be fruitfully exploited to emphasize interactions at different levels of granularity, in future works we will involve a domain expert in the analysis of results in order to evaluate qualitatively whether this idea appears confirmed by real biological findings.

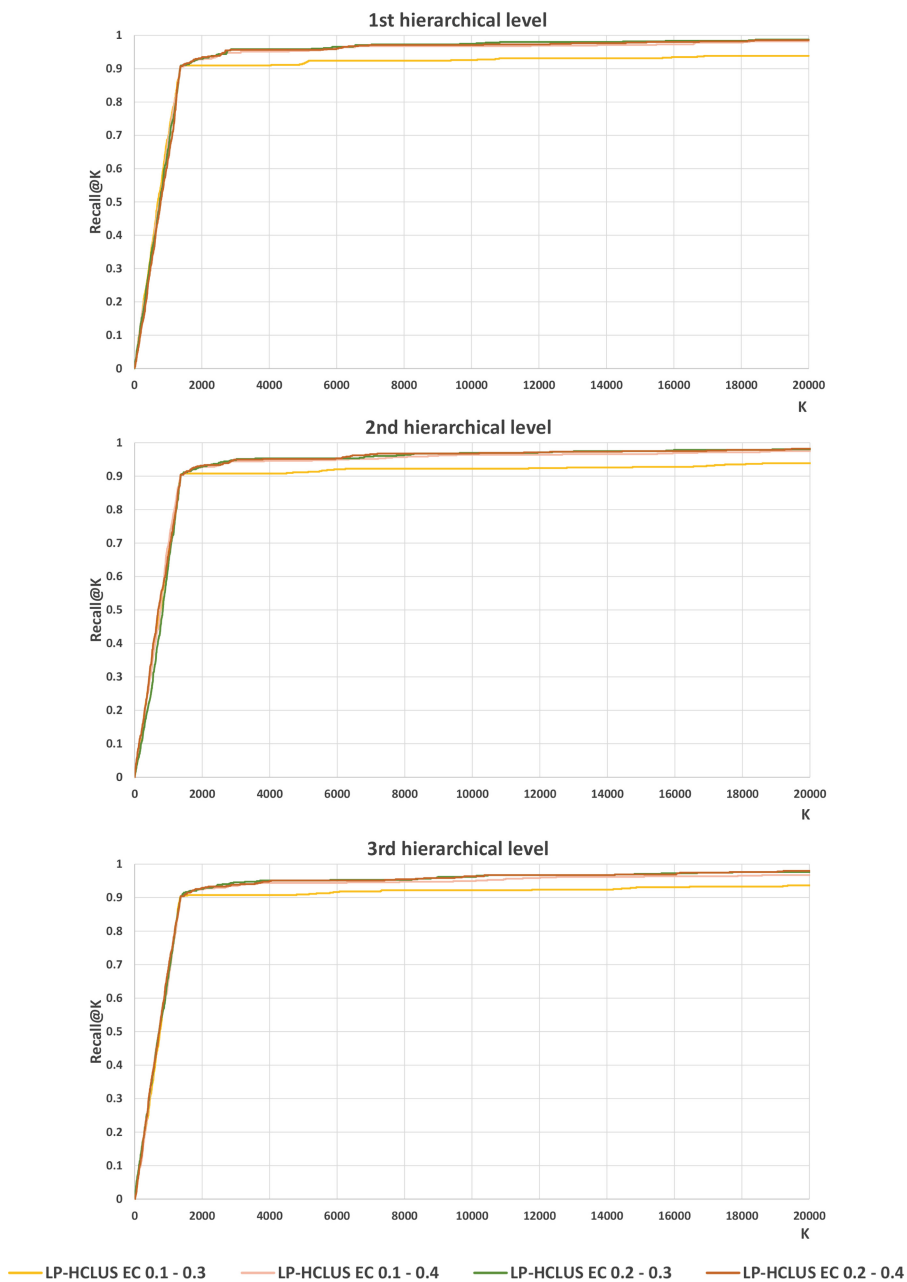


Fig. 5. Recall@K obtained by LP-HCLUS EC with different values of α and β .

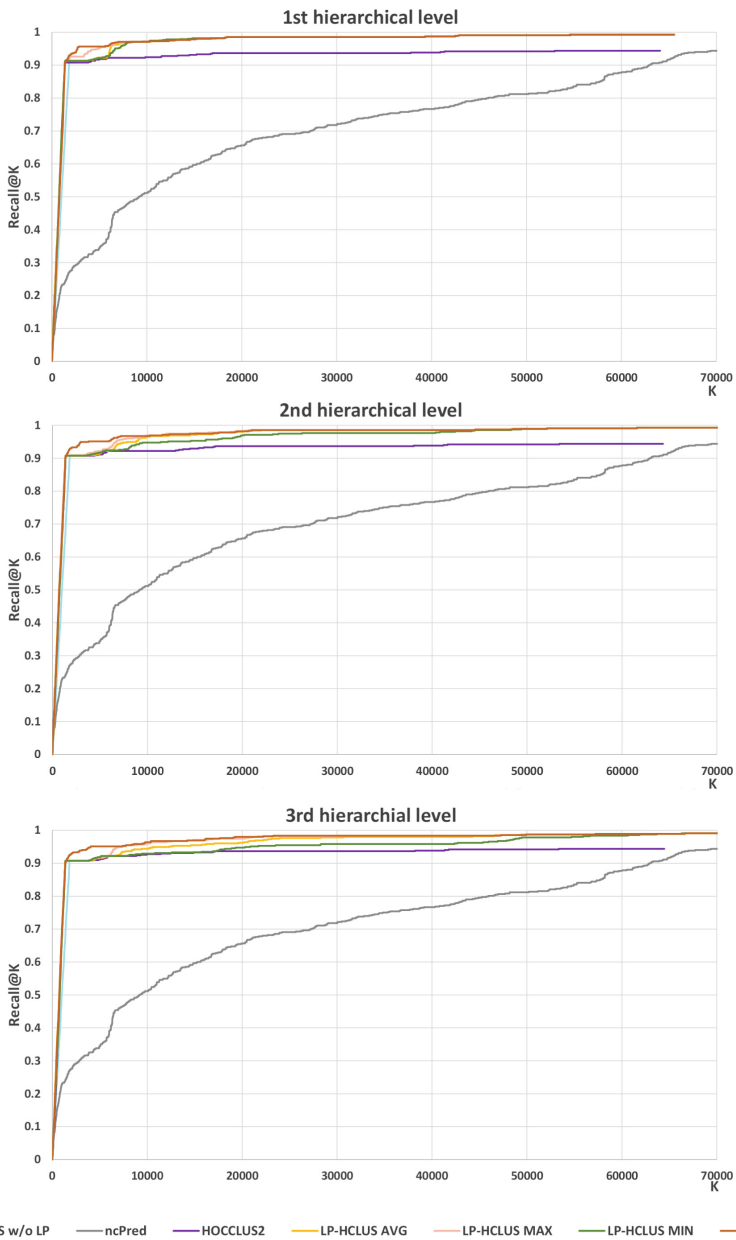


Fig. 6. *Recall@K* results obtained by LP-HCLUS ($\alpha = 0.2, \beta = 0.4$) and by the considered competitor methods.

Table 1. *Recall@K* obtained by LP-HCLUS and by the competitors for different values of *K*. Results obtained by HOCCLUS2 and LP-HCLUS have been collected for the first 3 levels of the hierarchies. For each value of *K*, the best result is highlighted in bold.

	Value of K									
	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
LP-HCLUS w/o LP	0.242	0.503	0.750	0.908	0.908	0.908	0.908	0.908	0.910	0.913
ncPred	0.154	0.228	0.248	0.275	0.293	0.304	0.316	0.325	0.336	0.344
11_HOCCLUS2	0.329	0.658	0.908	0.908	0.908	0.908	0.908	0.910	0.913	0.919
11_LP-HCLUS AVG	0.349	0.665	0.913	0.913	0.913	0.913	0.913	0.915	0.919	0.922
11_LP-HCLUS MAX	0.340	0.644	0.915	0.926	0.926	0.926	0.933	0.942	0.948	0.951
11_LP-HCLUS MIN	0.331	0.683	0.913	0.913	0.913	0.913	0.913	0.915	0.919	0.922
11_LP-HCLUS EC	0.311	0.629	0.913	0.933	0.939	0.957	0.957	0.957	0.957	0.957
12_HOCCLUS2	0.324	0.676	0.908	0.908	0.908	0.908	0.908	0.908	0.910	0.911
12_LP-HCLUS AVG	0.362	0.658	0.908	0.908	0.908	0.908	0.908	0.910	0.910	0.915
12_LP-HCLUS MAX	0.336	0.691	0.908	0.910	0.910	0.910	0.911	0.917	0.920	0.922
12_LP-HCLUS MIN	0.329	0.656	0.908	0.908	0.908	0.908	0.908	0.911	0.913	0.917
12_LP-HCLUS EC	0.349	0.662	0.910	0.931	0.933	0.949	0.949	0.951	0.951	0.951
13_HOCCLUS2	0.304	0.669	0.908	0.908	0.908	0.908	0.908	0.910	0.910	0.911
13_LP-HCLUS AVG	0.353	0.680	0.908	0.908	0.908	0.908	0.908	0.908	0.911	0.917
13_LP-HCLUS MAX	0.329	0.680	0.908	0.908	0.908	0.908	0.908	0.910	0.915	0.917
13_LP-HCLUS MIN	0.318	0.667	0.908	0.908	0.908	0.908	0.908	0.910	0.915	0.919
13_LP-HCLUS EC	0.358	0.689	0.910	0.930	0.933	0.939	0.940	0.949	0.951	0.951

4 Conclusions

In this work, we proposed the method LP-HCLUS, which is able to analyze heterogeneous biological networks in order to identify (possibly overlapping) hierarchically organized clusters and to exploit them to predict possibly unknown lncRNA-disease relationships. Such findings can be exploited for better understanding the role of lncRNAs in the development of human diseases. We also proposed the adoption of a specific strategy, based on evidence combination, to aggregate the different degrees of certainty when a new lncRNA-disease relationship is suggested by multiple clusters. Experiments performed on an integrated biological dataset showed that the proposed method, especially when adopting the strategy based on evidence combination, is able to outperform the methods HOCCLUS2 and ncPred, as well as a baseline approach. As future work, we intend to perform additional experiments on large-scale networks, possibly exploiting a distributed variant of the method proposed in this paper. Moreover, we will perform a qualitative evaluation, from a biological point of view, of the real contribution provided by our computational approach in the identification of lncRNA-disease relationships.



Acknowledgements. We would like to acknowledge the support of the European Commission through the projects MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data (Grant Number ICT-2013-612944) and TOREADOR - Trustworthy Model-aware Analytics Data Platform (Grant Number H2020-688797).

References

1. Alaimo, S., Giugno, R., Pulvirenti, A.: ncPred: ncRNA-disease association prediction through tripartite network-based inference. *Front. Bioeng. Biotechnol.* **2**, 71 (2014)
2. Bauer-Mehren, A., Rautschka, M., Sanz, F., Furlong, L.I.: DisGeNET: a cytoscape plugin to visualize, integrate, search and analyze gene-disease networks. *Bioinformatics* **26**(22), 2924–2926 (2010)
3. Cech, T., Steitz, J.: The noncoding RNA revolution—trashing old rules to forge new ones. *Cell* **157**(1), 77–94 (2014)
4. Ceci, M., Pio, G., Kuzmanovski, V., Dzeroski, S.: Semi-supervised multi-view learning for gene network reconstruction. *PLOS ONE* **10**(12), 1–27 (2015)
5. Chen, G., Wang, Z., Wang, D., Qiu, C., Liu, M., Chen, X., Zhang, Q., Yan, G., Cui, Q.: LncRNADisease: a database for long-non-coding RNA-associated diseases. *Nucleic Acids Res.* **41**(D1), D983–D986 (2013)
6. Han, J., Kamber, M., Pei, J.: *Data Mining: Concepts and Techniques*, 2nd edn. Morgan Kaufmann, San Francisco (2006)
7. Hayes, J., Peruzzi, P.P., Lawler, S.: MicroRNAs in cancer: biomarkers, functions and therapy. *Trends Mol. Med.* **20**(8), 460–469 (2014)
8. Helwak, A., Kudla, G., Dudnakova, T., Tollervey, D.: Mapping the human miRNA interactome by CLASH reveals frequent noncanonical binding. *Cell* **153**(3), 654–665 (2013)
9. Jiang, Q., Wang, Y., Hao, Y., Juan, L., Teng, M., Zhang, X., Li, M., Wang, G., Liu, Y.: miR2Disease: a manually curated database for microRNA deregulation in human disease. *Nucleic Acids Res.* **37**(suppl 1), D98–D104 (2009)
10. Lesmo, L., Saitta, L., Torasso, P.: Evidence combination in expert systems. *Int. J. Man-Mach. Stud.* **22**(3), 307–326 (1985)
11. Melissari, M.T., Grote, P.: Roles for long non-coding RNAs in physiology and disease. *Pflügers Archiv - Eur. J. Physiol.* **468**(6), 945–958 (2016)
12. Pio, G., Ceci, M., D’Elia, D., Loglisci, C., Malerba, D.: A novel biclustering algorithm for the discovery of meaningful biological correlations between microRNAs and their target genes. *BMC Bioinformatics* **14**(S-7), S8 (2013)
13. Pio, G., Ceci, M., Malerba, D., D’Elia, D.: ComiRNet: a web-based system for the analysis of miRNA-gene regulatory networks. *BMC Bioinformatics* **16**(9), S7 (2015)
14. Pio, G., Malerba, D., D’Elia, D., Ceci, M.: Integrating microRNA target predictions for the discovery of gene regulatory networks: a semi-supervised ensemble learning approach. *BMC Bioinformatics* **15**(1), S4 (2014)
15. Pio, G., Serafino, F., Malerba, D., Ceci, M.: Multi-type clustering and classification from heterogeneous networks. *Inf. Sci.* **425**, 107–126 (2018)
16. Yang, X., Gao, L., Guo, X., Shi, X., Wu, H., Song, F., et al.: A network based method for analysis of lncRNA-disease associations and prediction of lncRNAs implicated in diseases. *PLOS ONE* (2014)
17. Zadeh, L.: Fuzzy sets. *Inf. Control* **8**(3), 338–353 (1965)



Density Estimators for Positive-Unlabeled Learning

Teresa M. A. Basile^{2,3}, Nicola Di Mauro¹ , Floriana Esposito¹ ,
Stefano Ferilli¹, and Antonio Vergari¹

¹ Department of Computer Science, University of Bari “Aldo Moro”, Bari, Italy
nicola.dimauro@uniba.it

² Department of Physics, University of Bari “Aldo Moro”, Bari, Italy

³ National Institute for Nuclear Physics (INFN), Bari Division, Bari, Italy

Abstract. Positive-Unlabeled (PU) learning works by considering a set of positive samples, and a (usually larger) set of unlabeled ones. This challenging setting requires algorithms to cleverly exploit dependencies hidden in the unlabeled data in order to build models able to accurately discriminate between positive and negative samples. We propose to exploit probabilistic generative models to characterize the distribution of the positive samples, and to label as reliable negative samples those that are in the lowest density regions with respect to the positive ones. The overall framework is flexible enough to be applied to many domains by leveraging tools provided by years of research from the probabilistic generative model community. Results on several benchmark datasets show the performance and flexibility of the proposed approach.

1 Introduction

The classical supervised setting of statistical machine learning [14] aims at inducing models (classifiers) from training sets of labeled data in the form of samples (\mathbf{x}^i, y^i) i.i.d. drawn from an unknown joint probability distribution $p(\mathbf{X}, Y)$ over random variables (RVs) \mathbf{X} and Y , where Y is the *label*. For binary classification, i.e., $Y \in \{0, 1\}$, labels y^i are assumed to be modeled by a Bernoulli distribution and are associated to *positive* and *negative* samples \mathbf{x}^i .

While nowadays gathering and storing all kinds of data is easier and easier, having all these data perfectly and reliably labeled is unrealistic for several reasons, which makes classical approaches to learning classifiers inapplicable. First, the exponential rate at which data are produced contrasts the time required to produce high quality labels. Moreover, in many fields there are relatively few *labelers* effectively trained to produce reliable labels. Lastly, in many real-world domains it is sometimes unclear what should be considered as a negative sample, or the generation of negative samples is too expensive or just impossible. E.g., in process enactment, one would not waste time, money and resources to build a wrong item just for the purpose of showing how things are not to be done. Thus, the ability to learn predictive models in these scenarios may allow one to exploit the vast amount of data that are produced, saving precious time and resources.

In Positive-Unlabeled (PU) learning [8, 23], a set \mathcal{P} of positive samples, and a set \mathcal{U} of unlabeled samples—each of which may be positive or negative—are available at training time. So, discriminative information for the negative class must be found in unlabeled data. PU learning shares similarities with semi-supervised learning [26], one-class classification [28], and outlier detection [5]. Differently from the first, no negative samples are available at training time and yet it is required to learn a discriminator between the two classes, in contrast with the second. Additionally, PU learning is in opposition to the last which is usually performed in a transductive way to label unlabeled training data only.

The interest for PU learning is supported by its successful application in several domains [21, 35, 37]. PU learning approaches can be roughly grouped into *two-staged*—extracting a set of reliable negative samples (RN) from \mathcal{U} and then performing supervised learning—and *single-staged*—taking all samples in \mathcal{U} as negative. For the former, it becomes crucial to learn a metric that is able to discriminate among classes. However, each application domain needs a specific formulation for such a metric. Hence, ad hoc algorithmic solutions are often required to cope with different data representations [18, 37]. Few approaches have been proposed to deal with this categorical data [4, 18] in PU learning, due to their being quite challenging—there is no natural distance for them [19].

This work introduces *Generative Positive-Unlabeled* (GPU) learning, a novel two-staged approach to PU learning that aims to be general enough to support very different application domains¹. It estimates the marginal distribution $p_{\mathcal{P}}(\mathbf{X}|Y = 1)$ of the positive samples in \mathcal{P} via a generative model, and then performs inference on such a distribution to select a set of reliable negative samples from \mathcal{U} . The modeled probability density implicitly defines a metric space among samples, and we assume negative ones to be concentrated where positive ones are less likely. Generative models such as Probabilistic Graphical Models (PGMs) [20] have been extensively studied in the literature and offer a powerful formalism to deal with complex probability distributions over continuous, categorical, or even structured data [33]. Dealing with a particular domain translates into choosing a suitable PGM from a consolidated research field. More generally, given a PGM learned as a density estimator in a certain domain, we exploit it as a negative sample extractor for partially labeled data. Albeit GPU can deal with different data representations, here we focus on categorical data, which are handled natively by PGMs. We compared GPU on real data to several PU learners that have proven to be effective on categorical data.

The paper is organized as follows: in the next section we provide a brief review of the literature about PU learning. In Sect. 3 we introduce and discuss our GPU approach, while the experimental setting and the experiment results are presented in Sect. 4. Conclusions are drawn in Sect. 5.

¹ This paper is an extended version of [2] presented at the International Workshop NFMCP held in conjunction with ECML/PKDD 2017.

2 Related Works

PU learning has attracted a great deal of attentions in machine learning and data mining research. An extensively adopted approach to PU learning is based on a negative set construction process, that first identifies reliable negative samples from the unlabeled ones and, then, directly applies traditional classification methods. Alternative methods following this paradigm differ for how they implement these two steps.

Several proposals adopt distance-based approaches to identify negative samples, as the farthest unlabeled ones from positive samples. In [34], after selecting features statistically related to positive samples, the unlabeled set is partitioned into four sets (reliable/likely/weak negative and likely positive) based on the Euclidean distance. Successively, a multi-level samples learning technique, weighted SVMs, is exploited to build a classifier. The same approach of first identifying, characterizing and discriminating features for positive samples is adopted in [18], where a particular distance function previously designed by the authors is used to determine reliable negative samples; then, distance learning is applied twice—on the positive and reliable negative samples—and the resulting distances are used for k -NN classification.

After having theoretically shown that, under appropriate conditions, \mathcal{P} and \mathcal{U} provide sufficient information for learning, in [23] PU learning is posed as a constrained optimization problem. In such a setting, the set of reliable negative samples is selected by using a Naive Bayes (NB) classifier and EM. To the extreme, all the unlabeled samples are treated as negative samples in the NB classifier initially learned and successively used to extract the set of reliable negatives from unlabeled data [22]. The dataset so obtained is finally exploited to learn a classifier using SVM. The obtained augmented set, as representative of negatives samples, is exploited, along with positive ones, to compute the parameters of the NB classifier devoted to reliable negative samples identification. Finally, an EM-based algorithm is exploited to learn the predictive model.

A different policy is the weighted-based approach on unlabeled data exploited in [12]. The study shows that a classifier trained on positive and unlabeled samples is able to predict probabilities that differ by only a constant factor from the true conditional probabilities produced by a model trained on fully labeled positive and negative samples, provided that the labeled positive samples are chosen completely at random from all positive samples. This result is used in two different ways: learning from \mathcal{P} versus \mathcal{U} with adjustment of output probabilities finally assigned to unlabeled samples, and learning from \mathcal{P} and \mathcal{U} after double weighting of \mathcal{U} . The basic learning algorithm for each method is an SVM with a linear kernel whose outputs are post-processed into calibrated probabilities by fitting a one-dimensional logistic regression function.

Naive Bayes is the classifier extensively adopted for categorical data in the four methods proposed in [4], namely (Average) Positive Naive Bayes ((A)PNB), based on Naive Bayes, and (Average) Positive TAN ((A)PTAN), two variants of the Tree Augmented Naive Bayes model [13] able to deal with positive and unlabeled samples. The difference lies in the way the prior probability for the negative

class is estimated. For PNB and PTAN this probability is derived directly from the whole set of unlabeled samples while for APNB and APTAN the uncertainty is modeled by a Beta distribution.

The above survey shows that many works on PU learning [4, 18, 22, 23] have adopted the text categorization perspective, which is quite peculiar. Indeed, features are intrinsically categorical, there is a huge number of features compared to other settings, the representation of samples is very sparse, and there is a heavy impact of text pre-processing in setting up the classification problem. Others have faced biomedical problems [12, 34], where it is typical that databases specify which genes or proteins are related to some specific consequence, but this does not mean that all the others are unrelated to that consequence and, on the contrary, there is a strong interest in identifying which ones actually are [12].

As previously pointed out, although PU learning shares similarities to outlier detection [5] and to one-class classification [28], it shows difference from these settings in both the goal to fulfill and in the training set exploited, even in the case of probability density estimation techniques are used as solving strategies [15, 27, 29, 32]. Indeed, both methods aim at learning a model able to reject the new incoming samples using positive training data only. They do not require to learn a discriminator between the two classes and, hence, no effort to learn a model for the negative class is done. Intuitively, this type of approach is inferior because it ignores useful information that is present in the unlabeled samples.

3 Methodology

Let RVs be denoted by upper-case letters, e.g., X , and their values as the corresponding lower-case letters, e.g., $x \sim X$. We denote sets of RVs as \mathbf{X} , and their combined values as \mathbf{x} . When we refer to a joint probability distribution $\mathbf{p}(\mathbf{X})$ over RVs \mathbf{X} , we are either considering the joint probability density function for continuous RVs, or the probability mass function for discrete RVs, or a hybrid combination of both in hybrid domains [20, 33]. To denote a finite domain of a discrete RV X_j we introduce the following notation $\text{Val}(X_j) = \{x_j^k\}_{k=1}^K$. If \mathcal{D} is a set of samples over RVs \mathbf{X} , we indicate with $\mathbf{p}_{\mathcal{D}}(\mathbf{X})$ the real (unknown) probability distribution that generated the data, while if \mathcal{M} indicates a generative model, $\mathbf{p}_{\mathcal{M}}(\mathbf{X})$ refers to the probability distribution estimated by such a model on finite sample sets. Disambiguation is provided by context. Generally one wants the estimate $\mathbf{p}_{\mathcal{M}}(\mathbf{X})$ to be as close as possible to $\mathbf{p}_{\mathcal{D}}(\mathbf{X})$. A common way to measure this closeness is via the *log-likelihood* function [20], or one of its variants, defined as:

$$\ell_{\mathcal{D}}(\mathcal{M}) = \sum_{\mathbf{x}^i \in \mathcal{D}} \log \mathbf{p}_{\mathcal{M}}(\mathbf{x}^i).$$

In the classical PU learning setting, one has a training set $\mathcal{D} = \mathcal{P} \cup \mathcal{U}$ i.i.d. from $\mathbf{p}(\mathbf{X}, Y)$. Samples in \mathcal{P} are provided with a known positive class label, i.e., $\mathcal{P} = \{(\mathbf{x}^i, 1)\}_{i=1}^{m_{\mathcal{P}}} \sim \mathbf{p}_{\mathcal{P}}(\mathbf{X}|Y = 1)$. On the other hand, class information, i.e., labels, is not provided for samples in \mathcal{U} , i.e., $\mathcal{U} = \{\mathbf{x}^i\}_{i=1}^{m_{\mathcal{U}}} \sim \mathbf{p}_{\mathcal{U}}(\mathbf{X})$, where $\mathbf{p}_{\mathcal{U}}(\mathbf{X})$ is the marginal probability distribution w.r.t. $\mathbf{p}_{\mathcal{U}}(\mathbf{X}, Y)$. Let \mathcal{D}_0 (resp. \mathcal{D}_1)

denote the subset of all negative (resp. positive) samples in \mathcal{D} . The aim of PU learning is to build a discriminator model $f : \mathbf{X} \rightarrow Y$ from \mathcal{D} in order to make accurate predictions about the labels of unseen test data samples. Following [12], we assume that samples in \mathcal{P} are *selected completely at random* from all positive samples in \mathcal{D} , i.e., $\mathbf{p}_{\mathcal{P}}(\mathbf{X}|Y = 1) = \mathbf{p}_{\mathcal{D}}(\mathbf{X}|Y = 1)$.

3.1 Generative Models for PU Learning

Our proposed approach, Generative PU learning (GPU), falls in the category of two-staged methods for PU learning. First it extracts a set of reliable negative samples \mathcal{N} from \mathcal{U} , then \mathcal{N} is employed to perform supervised learning. In the following we detail our contribution to the first step, discussing possible approaches for the second one.

As usual in statistical machine learning, we assume $\mathbf{p}_{\mathcal{D}}$ to be modeled as a mixture of probability distributions for the positive and negative class, i.e.,

$$\mathbf{p}_{\mathcal{D}} = \sum_{y \in \{0,1\}} \mathbf{p}(Y = y)\mathbf{p}(\mathbf{X}|Y = y) = w_{\mathcal{D}_0}\mathbf{p}_{\mathcal{D}_0}(\mathbf{X}) + w_{\mathcal{D}_1}\mathbf{p}_{\mathcal{D}_1}(\mathbf{X}),$$

where $w_{\mathcal{D}_0}$ (resp. $w_{\mathcal{D}_1}$) denotes the marginal probabilities of the label w.r.t the negative (resp. positive) class and $\mathbf{p}_{\mathcal{D}_0}(\mathbf{X})$ (resp. $\mathbf{p}_{\mathcal{D}_1}(\mathbf{X})$) denotes the conditional probability of a sample w.r.t the negative (resp. positive) class. As already said, as it is common practice in PU learning [12], we assume that the positive samples in \mathcal{P} are highly representative for all positive samples in \mathcal{D}_1 . As an additional assumption, we consider the distribution generating \mathcal{D}_0 and \mathcal{D}_1 to be fairly *distinguishable* [1]. That is, we assume that high density regions of $\mathbf{p}_{\mathcal{D}_0}$ correspond to low density regions of $\mathbf{p}_{\mathcal{D}_1}$ and *vice versa*. While this assumption might seem too strict for real data, in practice, it is commonly adopted when performing unsupervised clustering (e.g., Gaussian densities must be separable in EM and K -means). As future research, we plan to investigate how to adapt GPU learning to more complex learning settings.

The high level idea behind our approach is the following. By correctly modeling the probability distribution of positive samples over RVs \mathbf{X} , one can model discriminative patterns among samples in the form of *probabilistic dependencies* among their RVs. If this is done accurately, then a metric space is implicitly defined, associating low probability regions to negative samples and high probability ones to positive samples. Similar ideas have also been successfully investigated in applications for anomalous or outlier training samples [27, 32]. Algorithm 1 illustrates the general schema of our proposed GPU approach. In order to estimate $\mathbf{p}_{\mathcal{P}}$ we fit a generative model, \mathcal{G} , over the RVs \mathbf{X} of the positive training set (line 3). We discuss the choice of such an estimator in Sect. 3.2. After that, we derive an empirical estimation of the less dense (i.e., less likely) regions by computing the point-wise log-likelihood of \mathcal{G} over the samples in \mathcal{U} . Based on this information we build a set of reliable negative samples, denoted as \mathcal{N} (line 7), to be exploited in the second stage of PU learning. As already stated, such a schema is general enough to be adapted to different data domains

Algorithm 1. LearnGPU(\mathcal{P}, \mathcal{U})

- 1: **Input:** a set $\mathcal{P} = \{(\mathbf{x}^i, 1)\}_{i=1}^{m_{\mathcal{P}}}$ of positive samples, and a set $\mathcal{U} = \{\mathbf{x}^i\}_{i=1}^{m_{\mathcal{U}}}$ of unlabeled samples over RVs $\mathbf{X} \cup \{Y\}$, with $\text{Val}(Y) = \{0, 1\}$.
 - 2: **Output:** a trained discriminative model learned on positive samples \mathcal{P} and reliable negative samples \mathcal{N} extracted from \mathcal{U}
 - 3: $\mathcal{G} \leftarrow \text{learnGenerativeModel}(\mathcal{P}, \mathbf{X})$ ▷ learn a generative model \mathcal{G} from \mathcal{P}
 - 4: $\mathcal{L} \leftarrow \{\log p_{\mathcal{G}}(\mathbf{x}^i) | \mathbf{x}^i \in \mathcal{U}\}$
 - 5: $\mathcal{N} \leftarrow \text{reliableNegativeSamples}(\mathcal{L}, \mathcal{P}, \mathcal{U})$
 - 6: $f \leftarrow \text{fitClassifier}(\mathcal{P}, \mathcal{N})$
 - 7: **return** f
-

by leveraging different density estimators. Moreover, by specifying algorithmic variants to build \mathcal{N} and the final discriminator f , one can improve its robustness and accuracy. We discuss such extensions in the following sections.

3.2 Bayesian Networks and Mixtures of Trees

A question arises on which generative model to employ. The main challenge in learning generative models is balancing the *representation expressiveness* of the learned models against the *cost of learning and performing inference* on them.

Probabilistic Graphical Models (PGMs), like Bayesian Networks (BNs) and Markov Networks (MNs), are able to model highly complex probability distributions and have been successfully employed as density estimators. However, exact inference with them is generally *intractable*. Since our GPU learning schema only requires the computation of complete evidence queries, employing BNs in GPU would lead to tractable inference to build \mathcal{N} .

Nevertheless, learning a complex model could still pose a challenge on very large datasets. Guaranteeing exact and tractable inference, a series of *tractable probabilistic models* (TPMs) have been recently proposed: either by restricting the expressiveness of PGMs by bounding their treewidth, or by exploiting local structures in a distribution. The limited expressive capabilities of TPMs, like mixtures of Bayesian trees (MT) [25] and Cutset Networks [9–11], or their ability to compile a high treewidth network into a deep probabilistic architecture, like Sum-Product Networks [31], allow for more efficient learning schemes.

In this work we evaluate GPU by exploiting both BNs and MTs to investigate how the model expressiveness affects the estimation of $\mathbf{p}_{\mathcal{P}}$ and therefore ultimately the accuracy of the learned discriminator (see Sect. 4). In the following we briefly review both models.

BNs are a PGM encoding a probability distribution by means of a directed acyclic graph and a set of weights, where nodes correspond to RVs and edges to dependencies among RVs. Given a set of n RVs \mathbf{X} , for each variable $X_i \in \mathbf{X}$, Pa_i denotes the set of parents of node X_i in the DAG. The structure of the BN \mathcal{G} , induces a factorization of the joint distribution into local factors, that is:

$$p_{\mathcal{G}}(\mathbf{X}) = \prod_{i=1}^n p(X_i | \text{Pa}_i).$$

Learning a BN corresponds to learning both the structure and the conditional probability distribution corresponding to each local factor from the data. Classical structure learning algorithms search in the space of BNs guided by a scoring function. On the other hand, parameter learning is obtained by maximum likelihood estimation.

Concerning mixtures of generative models, a very competitive density estimation algorithm is MT [25]. MT learns a mixture model \mathcal{M} whose distribution factorizes according to

$$p_{\mathcal{M}}(\mathbf{X}) = \sum_{i=1}^k \lambda_i p_{\mathcal{T}_i}(\mathbf{X}),$$

where the distributions $p_{\mathcal{T}_i}$, learned using the Chow-Liu algorithm [6], are the mixture components and $\lambda_i \geq 0$, with $\sum_{i=1}^k \lambda_i = 1$ are their coefficients. The Chow-Liu algorithm learns BNs with lower treewidth (i.e., nodes have at most one parent in the network), thus leading to efficient learning and inference time. In [25] the best components and weights are found as (local) likelihood maxima by using EM, with k fixed in advance.

3.3 Reliable Negative Sample Elicitation

After learning a generative model \mathcal{G} , the density estimation information \mathcal{G} provides can be exploited in several ways. The most straightforward one would be to impose a threshold hyperparameter θ such that each sample in \mathcal{U} whose log-likelihood $\log p_{\mathcal{G}}$ falls under θ can be added to \mathcal{N} . However, determining the best value for θ would require to perform additional hyperparameter optimization. To alleviate this issue we propose to implicitly compute it by building \mathcal{N} to comprise the $m_{\mathcal{P}} = |\mathcal{P}|$ samples in \mathcal{U} with the lowest log-likelihood score according to \mathcal{G} . In such a way we ensure that the resulting labeled set $\mathcal{P} \cup \mathcal{N}$ is balanced w.r.t. the positive and negative class. The risk of including positive samples into $\mathcal{P} \cup \mathcal{N}$ can be mitigated by adopting a robust classifier in the following supervised step, whose generalization ability on test data may also additionally benefit from the regularization capability of mis-specifying some sample labels. Lastly, we note how density information in the form of the finite set log-likelihoods can be directly incorporated into the construction of the classifier over $\mathcal{P} \cup \mathcal{N}$.

While we employ the likelihoods to select the most reliable negative samples from \mathcal{U} , they could also be used to select the most reliable positive samples instead. Adopting such a strategy, GPU can be turned into an iterative schema in which at each iteration \mathcal{P} is augmented with the samples belonging to the most dense regions. After a stopping criterion is met, \mathcal{N} can be built by collecting all the samples in \mathcal{U} not added to \mathcal{P} .

3.4 Supervised Classification Step

In principle, every supervised classifier can be employed in GPU after the set \mathcal{N} is constructed. In the empirical evaluation we provide in Sect. 4 we adopt the

regular implementation of Support Vector Machines (SVMs). Nevertheless, we now discuss other interesting variants for GPU learning. First, if one builds \mathcal{N} to be unbalanced w.r.t. \mathcal{P} , it would be possible to adopt the more robust variant of *biased SVMs* [17]. Alternatively, if one focuses on iteratively augmenting the set \mathcal{P} only with GPU, then 1-class SVMs [28] could be employed to derive a max-margin hypersphere for the positive class.

Additionally, the likelihoods associated to samples in \mathcal{U} could be interpreted as sample confidence weights. Approaches like that of [36] could be adopted to learn a *weighted classifier* over $\mathcal{P} \cup \mathcal{U}$ without the need to build \mathcal{N} either.

Lastly, our probabilistic generative approach for the first stage can be plugged in an *unsupervised clustering* approach for the second stage, as done with the EM algorithm in [23]. A principled end-to-end probabilistic formulation would allow estimating both $\mathbf{p}_{\mathcal{D}_0}$ and $\mathbf{p}_{\mathcal{D}_1}$ iteratively and jointly.

4 Experiments

In this section we empirically evaluate the proposed GPU approach, applying it to categorical data. We are interested in this kind of data because they are challenging for classical metric based approaches. Since there is no general consensus on how to build a metric to evaluate categorical data, ad-hoc solutions have been adopted on a domain-wise perspective [19], and only recently PU learning schemes have been devised for it [18]. On the other hand, PGMs have been extensively investigated for categorical data and estimating a probability distribution over discrete RVs is a consolidated practice for extracting new representations in a domain-agnostic unsupervised way [3, 16, 30]. As stated in the previous sections, adapting GPU to other domains reduces to selecting an appropriate generative toolbox from the probabilistic model literature. Specifically, we aim at answering the following research questions: **(Q1)** how does GPU compare to state-of-the-art PU learning approaches? **(Q2)** how does the quantity of available positive samples affect GPU learning? **(Q3)** how much does the choice of a generative model in estimating $\mathbf{p}_{\mathcal{P}}$ affect GPU’s performance?

4.1 Experimental Setting

We took 10 datasets publicly available on the UCI machine learning repository², derived 3 experimental settings for each, and ran 10-fold cross validations exactly as in [18]³. The three settings were generated by putting in \mathcal{P} 30%, 40%, and 50% labeled samples of the positive class respectively, and in \mathcal{U} the remaining positive samples plus all the negative ones. When the dataset does not describe a binary classification problem, the two heavily populated classes were considered. In our experiments, all numerical attributes were discretized into 10 equal-width bins. Detailed dataset statistics are reported in Table 1.

² <http://archive.ics.uci.edu/ml/>.

³ The datasets and settings used in [18] were kindly provided by Dino Ienco.

Table 1. Dataset statistics. #pos and #unl denote the number of positive and unlabeled samples respectively.

Dataset	#attributes	% pos						#test
		30		40		50		
		#pos	#unl	#pos	#unl	#pos	#unl	
Audiology	69	15	79	20	74	26	68	11
Breast-cancer	9	54	203	72	185	91	166	29
Chess	36	451	2425	601	2275	751	2125	320
Dermatology	34	30	136	40	126	50	116	19
Hepatitis	19	9	130	12	127	15	124	16
Lymph	18	17	111	22	106	28	100	14
Nursery	8	1166	6562	1555	6173	1944	5784	859
Pima	8	135	556	180	511	225	466	77
Soybean	35	25	140	33	132	42	123	18
Vote	16	72	319	96	295	120	271	44

We evaluate GPU by employing either BNs (GPU^{BN}) or MTs (GPU^{MT}) as generative models (see Sect. 3.2). BNs are learnt using the R package `bnlearn`⁴ (release 4.1.1). To learn their structure we employed the simple score-based hill-climbing algorithm. In order to avoid overfitting of the network to the positive samples, the following $K2$ scoring function [7] was adopted⁵:

$$\text{score}_{K2}(\mathcal{G} : \mathcal{P}) = \log p(\mathcal{G}) + \sum_{i=1}^n \sum_{j=1}^{q_i} \left(\log \left(\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \right) + \sum_{k=1}^{r_i} \log(N_{ijk}!) \right),$$

where $p(\mathcal{G})$ represents the prior probability of the network G over the n RVs X_i , r_i is the number of states of variable X_i , q_i is the number of possible configurations of the parent set Pa_i , N_{ijk} is the number of instances in the data where variable X_i takes value x_{ik} and the set of variables Pa_i takes value w_{ij} , N_{ij} is the number of instances in the data where the variables in Pa_i take their j -th configuration w_{ij} . Concerning parameter estimation, we set the imaginary sample size to 1. MTs are learnt using the `Libra` [24] toolkit⁶ (version 1.1.2). We imposed the number of components to be 10.

As the classifier for the supervised second stage, we adopt the commonly used SVMs⁷ with an RBF kernel as implemented in `scikit-learn`⁸. The penalty

⁴ <http://www.bnlearn.com/>.

⁵ The same set of experiments have been conducted using the likelihood as scoring function, leading to overfitted models with an overall result worst than that obtained using the $K2$ score.

⁶ <http://libra.cs.uoregon.edu/>.

⁷ For this stage only, categorical data is one-hot encoded.

⁸ <http://scikit-learn.org/>.

parameter C and the kernel coefficient γ have been optimized with a cross validation on the following grid $C \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ and $\gamma \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$.

We compared GPU with Positive Naive Bayes (PNB), Average Positive Naive Bayes (APNB), Positive TAN (PTAN), Average Positive TAN (APTAN) [4] and Pulce [18] with $k = 7$. See Sect. 2 for a description of these methods.

Source code, in Python and R, of the proposed approach and scripts to reproduce the results are available at <https://github.com/nicoladimauro/GPU>.

4.2 Results and Discussion

Performance on the test set was evaluated using the F1-score measure of the accuracy, defined as $F = 2PR/(P+R) = 2tp/(2tp+fp+fn)$, where P and R are, respectively, the precision and the recall obtained by the algorithm, and tp , fp and fn are, respectively, the true positive, false positive and false negative samples. Since the number of positive samples is much larger than that of negative ones, as in [18] we directed the computation of P , R , and F1-score to the negative samples, differently from their classical setting, i.e., as $F = 2tn/(2tn+fp+fn)$. In particular, since no information for the negative class is provided, correctly predicting negative samples should be somehow harder than focusing on the positive counterparts.

Overall results are reported in Tables 2 and 3. We may note that PTAN and APTAN never won against the other approaches, while the two GPU approaches won 73.3% of the times (53.3% of the times GPU^{BN} alone), and each GPU approach won more times than any competitor (GPU^{BN} more than doubled the number of wins of each competitor). The worst-performing dataset for GPU approaches, and the only one where they perform neatly worse than all other competitors, is ‘hepatitis’. This may indicate that for such a dataset the distributions of the negative and positive class are hard to estimate as very different densities. Concerning question Q1, therefore, we can say that both GPU^{BN} and GPU^{MT} are competitive to the current state-of-the-art for categorical data. On datasets on which GPU^{BN} does not win in all settings, it still performs comparably or better on settings with larger \mathcal{P} sets. Overall, increasing the size of \mathcal{P} improves the models’ accuracy in a consistent way. At the same time, on datasets where both GPU approaches are competitive, they improve over other methods even with only 30% positive samples available (Q2). Lastly, we observe that while GPU^{BN} generally outperforms GPU^{MT}, the latter is still comparable to Pulce (see average ranks, Table 2) and overall more accurate than all other methods. To answer question Q3, we can state that the greater expressiveness of BNs, allowing better modeling the probability distribution of the positive class, is fairly relevant for achieving better performances. Nevertheless, note that for both GPU^{BN} and GPU^{MT} we employed out-of-the-box PGMs and did not invest too much time optimizing the hyperparameters for their structure and weight learning algorithms. It is left for future work to explore how increasing a model complexity can degrade its performance, that is when too accurate probability distribution estimates can lead to overfitting.

Table 2. F1-score results over the 30 samples, comparing GPU^{BN} and GPU^{MT} against the competitors Pulce, PNB, APNB, PTAN, APTAN. The second column indicates the percentage of positive samples in \mathcal{P} .

Dataset	%	GPU ^{BN}	GPU ^{MT}	Pulce	PNB	APNB	PTAN	APTAN
Audiology	30	0.839	0.902	0.745	0.68	0.7	0.66	0.66
Audiology	40	0.879	0.804	0.846	0.75	0.74	0.71	0.66
Audiology	50	0.980	0.991	0.899	0.80	0.80	0.78	0.71
Breast-cancer	30	0.475	0.450	0.534	0.40	0.39	0.43	0.43
Breast-cancer	40	0.483	0.513	0.438	0.42	0.40	0.43	0.45
Breast-cancer	50	0.517	0.535	0.443	0.42	0.41	0.44	0.44
Chess	30	0.689	0.663	0.696	0.58	0.64	0.59	0.64
Chess	40	0.691	0.665	0.688	0.58	0.64	0.60	0.64
Chess	50	0.773	0.650	0.655	0.58	0.64	0.60	0.64
Dermatology	30	1.000	0.834	0.992	0.57	0.57	0.57	0.56
Dermatology	40	1.000	0.836	0.992	0.57	0.58	0.57	0.57
Dermatology	50	0.992	0.951	0.992	0.59	0.60	0.57	0.58
Hepatitis	30	0.822	0.665	0.843	0.87	0.87	0.85	0.86
Hepatitis	40	0.778	0.654	0.873	0.88	0.88	0.85	0.85
Hepatitis	50	0.764	0.742	0.855	0.88	0.88	0.86	0.85
Lymph	30	0.827	0.782	0.851	0.84	0.85	0.79	0.84
Lymph	40	0.825	0.795	0.827	0.84	0.83	0.79	0.81
Lymph	50	0.824	0.835	0.814	0.86	0.87	0.81	0.82
Nursery	30	0.809	0.761	0.739	0.65	0.65	0.56	0.50
Nursery	40	1.000	0.762	0.773	0.69	0.69	0.61	0.56
Nursery	50	0.960	0.779	0.807	0.69	0.70	0.74	0.44
Pima	30	0.588	0.576	0.532	0.49	0.50	0.50	0.50
Pima	40	0.568	0.593	0.547	0.49	0.50	0.50	0.51
Pima	50	0.609	0.605	0.528	0.49	0.51	0.50	0.52
Soybean	30	0.893	0.766	0.738	0.81	0.86	0.80	0.81
Soybean	40	0.883	0.852	0.767	0.86	0.86	0.84	0.83
Soybean	50	0.890	0.923	0.823	0.92	0.92	0.88	0.86
Vote	30	0.826	0.799	0.679	0.62	0.62	0.56	0.55
Vote	40	0.850	0.790	0.800	0.71	0.71	0.58	0.54
Vote	50	0.844	0.829	0.829	0.77	0.77	0.61	0.56
# wins		16	6	3	5	6	0	0
Avg. F1-score		0.796	0.743	0.751	0.677	0.686	0.653	0.643
30%		0.777	0.720	0.735	0.651	0.665	0.631	0.635
40%		0.796	0.727	0.755	0.679	0.683	0.648	0.642
50%		0.815	0.784	0.764	0.700	0.710	0.679	0.652
Avg. ranking		2.16	3.23	3.2	4.57	3.95	5.47	5.42

Table 3. Number of wins/ties among all the methods, the average of the number of wins for each method and its ranking in parenthesis.

	GPU ^{BN}	GPU ^{MT}	Pulce	PNB	APNB	PTAN	APTAN	Avg.
GPU ^{BN}	—	24/0	23/1	23/0	23/0	27/0	25/1	24.17 (1)
GPU ^{MT}	6/0	—	13/0	22/0	22/0	25/0	24/0	18.67 (3)
Pulce	6/0	17/0	—	22/0	22/0	25/0	24/0	19.33 (2)
PNB	7/0	8/0	8/0	—	5/12	18/2	18/3	10.67 (5)
APNB	7/0	8/0	8/0	13/12	—	23/3	21/4	13.33 (4)
PTAN	3/0	5/0	5/0	10/2	4/3	—	12/6	6.50 (7)
APTAN	4/0	6/0	6/0	9/3	5/4	12/6	—	7.00 (6)

Table 4. Number of positive samples incorrectly predicted as negative ones in the negative sample elicitation phase. In parenthesis the average number of errors for each fold over the cardinality of both \mathcal{P} and \mathcal{N} .

Dataset	30%		40%		50%	
Audiology	0.17	(2.6/15)	0.11	(2.3/20)	0.11	(2.8/26)
Breast-cancer	0.48	(25.7/54)	0.46	(33.3/72)	0.43	(38.8/91)
Chess	0.33	(148.6/451)	0.27	(162.4/601)	0.21	(159.7/751)
Dermatology	0.00	(0.0/30)	0.00	(0.0/40)	0.00	(0.0/50)
Hepatitis	0.19	(1.7/9)	0.00	(0.0/12)	0.11	(1.7/15)
Lymph	0.16	(2.8/17)	0.15	(3.4/22)	0.14	(3.8/28)
Nursery	0.00	(0.0/1166)	0.00	(0.0/1555)	0.03	(50.4/1944)
Pima	0.33	(44.4/135)	0.30	(53.6/180)	0.28	(64.0/225)
Soybean	0.14	(3.4/25)	0.15	(4.9/33)	0.10	(4.4/42)
Vote	0.30	(22.9/72)	0.22	(21.0/96)	0.23	(27.9/120)

As already said, in the reliable negative sample elicitation phase, the number of negative samples to be included in the set \mathcal{N} was set to the same number of positive samples available in \mathcal{P} , i.e., $|\mathcal{N}| = |\mathcal{P}|$. However, the generated set \mathcal{N} may contain some true positive samples that have been incorrectly predicted as negative ones. In order to quantify the accuracy of the proposed approach, Table 4 reports, for each dataset, the number of errors occurred in the negative elicitation step, when GPU^{BN} has been used as a density estimator. As we can see, the percentage of errors decreases as the percentage of positive samples in \mathcal{P} increases. For datasets like ‘breast-cancer’, the number of errors reaches 50% thus confirming the low accuracy in terms of F1-score obtained on this dataset. Anyway, also other competitors are not able to properly separate positive and negative samples on this dataset.

A concluding experiment has been done by varying the number of negative samples in \mathcal{N} as a percentage of the number of samples in \mathcal{P} , i.e., by setting

Table 5. Detailed F1-score results over the 30 samples obtained by GPU^{BN} by varying the percentage of the reliable negative samples added to \mathcal{N} . Last column reports Pulce results for comparison.

Dataset	GPU ^{BN}										Pulce
	%	Negative percentage									
	60%	70%	80%	90%	100%	110%	120%	130%	140%		
Audiology	30	0.830	0.832	0.812	0.834	0.839	0.627	0.857	0.857	0.880	0.745
Audiology	40	0.896	0.949	0.940	0.958	0.879	0.938	0.940	0.923	0.931	0.846
Audiology	50	1.000	0.991	0.989	0.971	0.980	0.989	0.966	0.957	0.942	0.899
Breast-cancer	30	0.478	0.422	0.426	0.410	0.475	0.485	0.497	0.464	0.450	0.534
Breast-cancer	40	0.421	0.416	0.409	0.470	0.483	0.494	0.492	0.500	0.452	0.438
Breast-cancer	50	0.457	0.486	0.454	0.508	0.517	0.522	0.497	0.481	0.476	0.443
Chess	30	0.604	0.629	0.644	0.668	0.689	0.693	0.693	0.700	0.700	0.696
Chess	40	0.601	0.637	0.671	0.675	0.691	0.703	0.724	0.731	0.740	0.688
Chess	50	0.623	0.645	0.701	0.739	0.773	0.777	0.788	0.786	0.783	0.655
Dermatology	30	0.992	0.992	0.992	0.992	1.000	0.992	0.992	0.992	0.992	0.992
Dermatology	40	0.992	0.992	0.992	0.992	1.000	1.000	1.000	1.000	1.000	0.992
Dermatology	50	0.992	0.992	0.992	0.992	0.992	0.992	1.000	1.000	0.993	0.992
Hepatitis	30	0.336	0.620	0.605	0.513	0.822	0.822	0.862	0.842	0.876	0.843
Hepatitis	40	0.562	0.611	0.678	0.741	0.778	0.825	0.823	0.871	0.839	0.873
Hepatitis	50	0.679	0.684	0.730	0.695	0.764	0.871	0.859	0.898	0.891	0.855
Lymph	30	0.792	0.810	0.791	0.793	0.827	0.782	0.800	0.829	0.794	0.851
Lymph	40	0.757	0.772	0.781	0.842	0.825	0.823	0.838	0.819	0.849	0.827
Lymph	50	0.721	0.792	0.814	0.841	0.824	0.833	0.816	0.823	0.792	0.814
Nursery	30	0.799	0.810	0.816	0.819	0.809	0.810	0.817	0.818	0.809	0.739
Nursery	40	1.000	1.000	1.000	1.000	1.000	1.000	0.832	0.832	0.832	0.773
Nursery	50	1.000	1.000	1.000	1.000	0.960	1.000	1.000	1.000	1.000	0.807
Pima	30	0.481	0.515	0.535	0.545	0.588	0.543	0.562	0.565	0.567	0.532
Pima	40	0.504	0.509	0.531	0.574	0.568	0.600	0.590	0.607	0.620	0.547
Pima	50	0.519	0.532	0.552	0.603	0.609	0.620	0.613	0.623	0.610	0.528
Soybean	30	0.802	0.834	0.816	0.902	0.893	0.914	0.902	0.915	0.929	0.738
Soybean	40	0.830	0.846	0.893	0.902	0.883	0.915	0.924	0.925	0.926	0.767
Soybean	50	0.819	0.886	0.864	0.854	0.890	0.922	0.916	0.935	0.938	0.823
Vote	30	0.819	0.841	0.838	0.832	0.826	0.798	0.795	0.801	0.779	0.679
Vote	40	0.883	0.850	0.861	0.864	0.850	0.848	0.831	0.814	0.806	0.800
Vote	50	0.892	0.918	0.888	0.854	0.844	0.845	0.846	0.791	0.824	0.829
# wins		12	14	16	20	22	21	25	23	24	
Avg. F1-score		0.736	0.760	0.767	0.779	0.796	0.799	0.802	0.803	0.801	0.751

$|\mathcal{N}| = \alpha|\mathcal{P}|$. Table 5 reports the results adopting GPU^{BN} as a density estimator for $\alpha \in \{0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4\}$. It is possible to see that for $\alpha = 1.2$ the number of wins of the proposed approach over Pulce increases to 25.

5 Conclusions

In Positive-Unlabeled (PU) learning only positive samples are labeled at training time. PU learning requires algorithms to cleverly exploit dependencies hidden in the data in order to build models able to discriminate between positive and negative samples. In this paper, we proposed to exploit probabilistic generative models for PU learning by characterizing the density distribution for the positive class. The overall GPU framework is flexible enough to be applied on many domains by leveraging tools provided by PGMs. Results on several benchmark datasets empirically confirmed the validity of our new proposed approach.

References

1. Balasubramanian, V.: MDL, Bayesian inference, and the geometry of the space of probability distributions. In: Grünwald, P.D., Myung, I.J., Pitt, M.A. (eds.) *Advances in Minimum Description Length: Theory and Applications*, pp. 81–98. MIT Press, Cambridge (2005)
2. Basile, T., Mauro, N.D., Esposito, F., Ferilli, S., Vergari, A.: Generative probabilistic models for positive-unlabeled learning. In: *Workshop on NFMCP Held with ECML/PKDD (2017)*
3. Bengio, Y., Courville, A.C., Vincent, P.: Unsupervised feature learning and deep learning: a review and new perspectives. *CoRR abs/1206.5538* (2012)
4. Calvo, B., Larraaga, P., Lozano, J.A.: Learning bayesian classifiers from positive and unlabeled examples. *Pattern Recogn. Lett.* **28**(16), 2375–2384 (2007)
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. *ACM Comput. Surv.* **41**(3), 15:1–15:58 (2009)
6. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theor.* **14**, 462–467 (1968)
7. Cooper, G.F., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. *Mach. Learn.* **9**(4), 309–347 (1992)
8. De Comité, F., Denis, F., Gilleron, R., Letouzey, F.: Positive and unlabeled examples help learning. In: Watanabe, O., Yokomori, T. (eds.) *ALT 1999. LNCS (LNAI)*, vol. 1720, pp. 219–230. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-46769-6_18
9. Di Mauro, N., Vergari, A., Basile, T.M.A., Esposito, F.: Fast and accurate density estimation with extremely randomized cutset networks. In: Ceci, M., Hollmén, J., Todorovski, L., Vens, C., Džeroski, S. (eds.) *ECML PKDD 2017. LNCS (LNAI)*, vol. 10534, pp. 203–219. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71249-9_13
10. Di Mauro, N., Vergari, A., Basile, T.M.A.: Learning Bayesian random cutset forests. In: Esposito, F., Pivert, O., Hacid, M.-S., Raš, Z.W., Ferilli, S. (eds.) *ISMIS 2015. LNCS (LNAI)*, vol. 9384, pp. 122–132. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25252-0_13
11. Di Mauro, N., Vergari, A., Esposito, F.: Learning accurate cutset networks by exploiting decomposability. In: Gavanelli, M., Lamma, E., Riguzzi, F. (eds.) *AI*IA 2015. LNCS (LNAI)*, vol. 9336, pp. 221–232. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24309-2_17
12. Elkan, C., Noto, K.: Learning classifiers from only positive and unlabeled data. In: *KDD*, pp. 213–220 (2008)

13. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. *Mach. Learn.* **29**(2–3), 131–163 (1997)
14. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning*. Springer, New York (2009). <https://doi.org/10.1007/978-0-387-84858-7>
15. Hempstalk, K., Frank, E., Witten, I.H.: One-class classification by combining density and class probability estimation. In: Daelemans, W., Goethals, B., Morik, K. (eds.) *ECML PKDD 2008*. LNCS (LNAI), vol. 5211, pp. 505–519. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87479-9_51
16. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* **313**, 504–507 (2006)
17. Hoi, C.H., Chan, C.H., Huang, K., Lyu, M.R., King, I.: Biased support vector machine for relevance feedback in image retrieval. In: *IJCNN*, pp. 3189–3194 (2004)
18. Ienco, D., Pensa, R.G.: Positive and unlabeled learning in categorical data. *Neurocomputing* **196**, 113–124 (2016)
19. Ienco, D., Pensa, R.G., Meo, R.: From context to distance: learning dissimilarity for categorical data clustering. *TKDD* **6**(1), 1:1–1:25 (2012)
20. Koller, D., Friedman, N.: *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge (2009)
21. Li, H., Chen, Z., Liu, B., Wei, X., Shao, J.: Spotting fake reviews via collective positive-unlabeled learning. In: *ICDM*, pp. 899–904 (2014)
22. Liu, B., Dai, Y., Li, X., Lee, W.S., Yu, P.S.: Building text classifiers using positive and unlabeled examples. In: *ICDM*, pp. 179–188 (2003)
23. Liu, B., Lee, W.S., Yu, P.S., Li, X.: Partially supervised classification of text documents. In: *ICML*, pp. 387–394 (2002)
24. Lowd, D., Rooshenas, A.: The Libra toolkit for probabilistic models. *CoRR* abs/1504.00110 (2015)
25. Meila, M., Jordan, M.I.: Learning with mixtures of trees. *JMLR* **1**, 1–48 (2000)
26. du Plessis, M.C., Sugiyama, M.: Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Netw.* **50**, 110–119 (2014)
27. Riahi, F., Schulte, O., Li, Q.: A proposal for statistical outlier detection in relational structures. In: *SRAI AAAI Workshop* (2014)
28. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. *Neural Comput.* **13**(7), 1443–1471 (2001)
29. Tax, D.M.J., Duin, R.P.W.: Support vector data description. *Mach. Learn.* **54**(1), 45–66 (2004)
30. Vergari, A., Di Mauro, N., Esposito, F.: Visualizing and understanding sum-product networks. *CoRR* abs/1608.08266 (2016)
31. Vergari, A., Di Mauro, N., Esposito, F.: Simplifying, regularizing and strengthening sum-product network structure learning. In: Appice, A., Rodrigues, P.P., Santos Costa, V., Gama, J., Jorge, A., Soares, C. (eds.) *ECML PKDD 2015*. LNCS (LNAI), vol. 9285, pp. 343–358. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23525-7_21
32. Xu, J., Shelton, C.R.: Intrusion detection using continuous time Bayesian networks. *J. Artif. Int. Res.* **39**(1), 745–774 (2010)
33. Yang, E., Baker, Y., Ravikumar, P., Allen, G., Liu, Z.: Mixed graphical models via exponential families. In: *AISTATS*, pp. 1042–1050 (2014)
34. Yang, P., Li, X.L., Mei, J.P., Kwok, C.K., Ng, S.K.: Positive-unlabeled learning for disease gene identification. *Bioinformatics* **28**, 2640–2647 (2012)
35. Zhao, Y., Kong, X., Philip, S.Y.: Positive and unlabeled learning for graph classification. In: *ICDM*, pp. 962–971 (2011)

36. Zhou, J., Pan, S., Mao, Q., Tsang, I.: Multi-view positive and unlabeled learning. In: ACML, pp. 555–570 (2012)
37. Zhou, K., Gui-Rong, X., Yang, Q., Yu, Y.: Learning with positive and unlabeled examples using topic-sensitive PLSA. TKDE **22**(1), 46–58 (2010)



Combinatorial Optimization Algorithms to Mine a Sub-Matrix of Maximal Sum

Vincent Branders^(✉), Pierre Schaus, and Pierre Dupont

ICTEAM/INGI, Machine Learning Group,
Université catholique de Louvain,
Place Sainte Barbe 2, 1348 Louvain-la-Neuve, Belgium
{vincent.branders,pierre.schaus,pierre.dupont}@ucclouvain.be

Abstract. Biclustering techniques have been widely used to identify homogeneous subgroups within large data matrices, such as subsets of genes similarly expressed across subsets of patients. Mining a *max-sum sub-matrix* is a related but distinct problem for which one looks for a (non-necessarily contiguous) rectangular sub-matrix with a maximal sum of its entries. Le Van et al. [7] already illustrated its applicability to gene expression analysis and addressed it with a constraint programming (CP) approach combined with large neighborhood search (LNS). In this work, we exhibit some key properties of this \mathcal{NP} -hard problem and define a bounding function such that larger problems can be solved in reasonable time. The use of these properties results in an improved CP-LNS implementation evaluated here. Two additional algorithms are also proposed in order to exploit the highlighted characteristics of the problem: a CP approach with a global constraint (CPGC) and a mixed integer linear programming (MILP). Practical experiments conducted both on synthetic and real gene expression data exhibit the characteristics of these approaches and their relative benefits over the CP-LNS method. Overall, the CPGC approach tends to be the fastest to produce a good solution. Yet, the MILP formulation is arguably the easiest to formulate and can also be competitive.

1 Introduction

Gene expression data is typically represented as a large matrix of gene expression levels across various samples. The study of such data is a valuable tool to improve the understanding of the underlying biological processes. For example, biomarker discovery aims at finding indicators of a disease or the physiological state of patients. This problem can be addressed with clustering techniques which perform a grouping of one dimension, either the rows or the columns of the original matrix. Yet it is known that breast cancer, for example, exhibits distinct subtypes [12, 13]. In other words, specific genes exhibit activation patterns only in a specific group of patients. Biclustering techniques, or co-clustering, identify specific subsets of rows and of columns which jointly form homogeneous entries [9].

In the present work, we focus on a related but different mining task. One looks in particular for subsets of rows and of columns with globally high values. In the context of gene expression analysis, the objective is to find a subset of genes which are relatively highly expressed among a subset of patients, even though some entries might depart from this pattern. With several thousands genes and hundreds of patients, we are mostly interested in the production of an algorithm that can deal with large matrices. Formally, one looks for a rectangular, and non necessarily contiguous, sub-matrix of a large matrix with a maximal sum of the selected entries. An illustrative example is provided in Fig. 1. The sum of the sub-matrix defined by the subset of rows $\{i_1, i_2, i_4, i_5\}$ and the subset of columns $\{j_2, j_4, j_5, j_6\}$ is 27.3 and is maximum. It can not be increased by the addition or the exclusion of any other row or column.

This *max-sum sub-matrix* problem is closely related to the *maximal ranked tile* mining problem studied by Le Van et al. [7]. In the later case, prior to the search of a sub-matrix, each matrix entry is replaced by its rank across its particular row. In other words, the maximal ranked tile mining is equivalent to the max-sum sub-matrix for which the matrix entries are discrete ranks along the rows. Since the combinatorial optimization algorithms to solve such problems are actually not specific to discrete entries we address here the slightly more general setting of continuous entries.

Our main contributions are (1) the study of the *max-sum sub-matrix* problem while exhibiting some of its key properties and the definition of an upper bound easy to compute in order to speed up the search for a solution; (2) the implementation of two additional algorithms making use of these properties: a CP approach with a global constraint (CPGC) and mixed integer linear programming (MILP); (3) practical experiments conducted both on synthetic and real gene expression data showing that the CP-LNS method can be largely outperformed; (4) the study of the relative benefits of the proposed methods across various problem instances.

2 Problem

2.1 Statement

Definition 1 (The Max-Sum Sub-Matrix Problem). *Given a matrix $\mathcal{M} \in \mathbb{R}^{m \times n}$ consisting of m rows and n columns, let $\mathcal{R} = \{1, \dots, m\}$ and $\mathcal{C} = \{1, \dots, n\}$ be index sets for rows and for columns respectively, find the max-sum sub-matrix (I^*, J^*) , with $I^* \subseteq \mathcal{R}$ and $J^* \subseteq \mathcal{C}$, such that:*

$$(I^*, J^*) = \operatorname{argmax}_{I \subseteq \mathcal{R}, J \subseteq \mathcal{C}} f(I, J) = \operatorname{argmax}_{I \subseteq \mathcal{R}, J \subseteq \mathcal{C}} \sum_{i \in I, j \in J} \mathcal{M}_{i,j}. \quad (1)$$

The *objective function* $f(I, J)$ is the sum of the entries of a sub-matrix (I, J) . The maximization term rewards, respectively penalizes, matrix entries with positive, respectively negative, values. One only considers matrices with positive and negative entries. Otherwise the optimal solution is trivially identified.

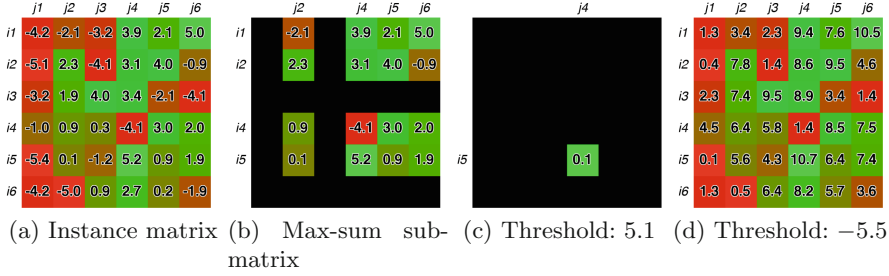


Fig. 1. (a): Instance matrix. (b): Associated sub-matrix of maximal sum. (c) and (d): Sub-matrix of maximal sum obtained for different thresholds applied on the instance matrix.

This formulation implicitly assumes that there is a threshold equal to zero to consider entries as potentially relevant or informative.

A different threshold can be considered by subtracting some constant to all matrix entries. This allows to control the size of the optimal sub-matrix. Figures 1c and d illustrates solutions for a threshold of 5.1 and -5.5 , respectively.

The max-sum sub-matrix problem is \mathcal{NP} -hard from a simple reduction¹ of the Maximum edge Weight Biclique Problem (MWBP) [4].

2.2 Explicit Versus Implicit Search Space

For a defined subset of columns J , the objective function can be formulated as $f(I, J) = \sum_{i \in I} r_i$ with r_i being the contribution of the row i :

$$r_i = \sum_{j \in J} \mathcal{M}_{i,j}. \quad (2)$$

This is important as the actual search can be limited to one dimension through independent computation of the contributions along the other dimension. Indeed, for any of the two dimensions being fixed, optimization along the other dimension is straightforward since it amounts to select only the subset of entries whose contribution is positive. For a fixed subset of columns $J \subseteq \mathcal{C}$, the subset of rows $I_J^* \subseteq \mathcal{R}$ that maximizes the objective value is identified as:

$$I_J^* = \operatorname{argmax}_{I \subseteq \mathcal{R}} \sum_{i \in I, j \in J} \mathcal{M}_{i,j} = \{i \in \mathcal{R} \mid r_i \geq 0\}. \quad (3)$$

In the gene expression analysis context, with order(s) of magnitude more rows (the genes) than columns (the samples), one typically searches for a subset of columns and selects the associated optimal subset of rows.

¹ Essentially by considering the rows and columns of the matrix as the two sets of nodes of a bipartite graph.

The search space of the max-sum sub-matrix problem contains $2^{|\mathcal{R}|} \times 2^{|\mathcal{C}|}$ *feasible solutions* that are rectangular sub-matrices ($I \subseteq \mathcal{R}, J \subseteq \mathcal{C}$) of the original matrix. Thanks to the independent contribution along one dimension, the number of feasible solutions to be explicitly evaluated is thus reduced to $2^{|\mathcal{C}|} \times |\mathcal{R}|$.

2.3 Related Work

Biclustering techniques address the problem of finding sub-matrices that satisfy some definition of homogeneity since entries grouped together into biclusters typically have similar values. A comprehensive review can be found in [9]. There is no assumption of homogeneity in the max-sum sub-matrix problem but rather one looks for a rectangular sub-matrix with an overall maximal sum. The difference is illustrated in Fig. 1b where a highly negative entry -4.1 in (i_4, j_4) is selected. This results from the sums of the selected entries in i_4 and in j_4 which contribute positively to the maximal sum. In the biclustering context, any entry that differs from entries of a bicluster or from entries in its row or its column is not expected to be selected. Cohesive biclusters [14, 15], with high average values, are built by aggregating entries that are higher than a certain threshold such that the average value of the bicluster is higher than a second threshold. Then all entries must be higher than the first threshold while in the max-sum sub-matrix problem there is no expected minimum value for an entry to be selected. Biclustering approaches commonly identify multiple biclusters. Many algorithms iteratively identify a single bicluster and subsequently mask it [3, 9, 16]. Yang et al. [20] proposed a global alternative to these local optimal decisions. Designing a dedicated approach to the identification of multiple sub-matrices of maximal sum is part of our future research.

The maximum (contiguous) subarray problem introduced in [2] identifies a subarray of maximal sum from an array. For a one-dimensional array, this problem can be solved in linear time by Kadane’s algorithm [2]. Cubic and sub-cubic time complexity algorithms have been proposed in the two-dimensional case [2, 18, 19]. This problem is however simpler than the max-sum sub-matrix since the selected sub-matrix is constrained to be formed of contiguous rows and contiguous columns from the original matrix.

The maximum ranked tile mining problem has been introduced in [7]. As discussed in Sect. 1, this is a special case of the max-sum sub-matrix problem for which the matrix entries are discrete ranks, corresponding to a permutation of column indices on each row. While the discrete ranking is an important characteristic of the maximum ranked tile mining problem, the associated constraint programming solution does not require discrete entries nor benefit from it. In this work, we present improved and new optimization approaches to solve the problem with arbitrary continuous entries.

Subgroup discovery is a data mining technique which extracts classification rules with respect to a target variable [1, 6]. It departs from the standard learning of a

classifier as the extracted rules are not necessarily intended to cover all possible instances. Besides, such technique focuses on the interpretability of the classification rules rather than the generalization capability to classify new instances. Mining a max-sum sub-matrix is somewhat similar to subgroup discovery but there is no supervision by a specific target class variable. The hotspot detection problem, which can be considered as a particular form of subgroup discovery, aims at identifying subgroups that are unexpected with respect to some baseline information [5]. In the max-sum sub-matrix problem, one does consider globally high values without any baseline distribution of the data.

3 Optimization Approaches

3.1 Boolean Decision Vectors

The max-sum sub-matrix problem can be modeled with two vectors of boolean decision variables: $T = (T_1, \dots, T_m)$ for the rows and $U = (U_1, \dots, U_n)$ for the columns with $T_i \in \{0, 1\}$ and $U_j \in \{0, 1\}$. A sub-matrix (I, J) is defined by $I = \{i \in \mathcal{R} \mid T_i = 1\}$ and $J = \{j \in \mathcal{C} \mid U_j = 1\}$. The problem consists in assigning a value to each variable of T and U . Let us denote by $U^1 = \{j \in \mathcal{C} \mid U_j = 1\}$ the selected columns, $U^0 = \{j \in \mathcal{C} \mid U_j = 0\}$ the unselected ones and by $U^? = \{j \in \mathcal{C} \mid U_j = \{0, 1\}\}$ the undecided ones.

3.2 Constraint Programming

Constraint Programming (CP) is a flexible programming paradigm that is capable of solving optimization problems. As a declarative approach, it only requires to model the problem and, by using existing solvers, let it search and find solutions. A model is defined as a constraint satisfaction problem $CSP = (V, D, C)$ where V is the set of variables, D is their respective domains, and C is a set of constraints defined over the variables. A feasible solution is an assignment of the variables to values of their domains such that all constraints are satisfied.

Constraints are exploited to iteratively reduce the domains of variables. Such *constraint propagation* reduces the number of variable assignments to consider. Once all unfeasible values are removed from the domains of variables, the *fix-point* of the propagation is reached. Then the solver selects a variable $X \in V$ that is unbound and recursively calls the solver while assigning a value to this variable. Through exploration of a depth-first-search tree (DFS), the solver either reaches a solution or backtracks when the domain of variables becomes empty.

Efficient backtracking is achieved through *trailing*, a state management strategy that facilitates the restoration of the computation state to an earlier version, effectively *undoing* changes that were imposed since then. The *trail* exposes two methods: `pushState` and `popState` to respectively time-stamp the current state and restore it. Its implementation is captured in terms of two simple stacks. The first stack holds entries to undo, the second one holds the frame sizes stacked between two consecutive call to `pushState`. Trailing enables the design of reversible objects defined on the trail.

In the rest of this section, the original model proposed by Le Van et al. [7] ($CP-LNS_0$) is presented, as well as an improved version ($CP-LNS$). Then a new constraint programming formulation is proposed ($CPGC$).

In $CP-LNS_0$, the tree reaches a leaf (or feasible solution) as soon as all rows and columns variables are bound. Each row and each column is associated to a reified constraint that ensures its selection if the sum of the entries along the other selected dimensions (respectively columns or rows) is positive. The LNS strategy is implemented as follows: after a given number of backtracking, the constraints on half of the columns variables of the best solution found so far are removed and the search restarts in another region. LNS may improve the time to identify a good solution at the cost of losing the possibility to prove optimality since the search is no longer complete.

In $CP-LNS$, by virtue of the implicit search space property (see Sect. 2.2), the tree reaches a leaf as soon as all column variables are bound. The contribution of all rows are computed afterwards. The objective function is computed as the sum of positive rows contributions. Similarly to $CP-LNS_0$, after some backtracking, half of the constraints, here only on the columns variables, are relaxed and a different region of the search space is explored.

CP-LNS₀. The max-sum sub-matrix problem has been modeled in [7] as:

$$\text{maximize } \sum_{i \in \mathcal{R}, j \in \mathcal{C}} T_i \times U_j \times \mathcal{M}_{i,j}, \quad (4)$$

$$\forall i \in \mathcal{R} : T_i = 1 \Leftrightarrow \sum_{j \in \mathcal{C}} U_j \times \mathcal{M}_{i,j} \geq 0, \quad (5)$$

$$\forall j \in \mathcal{C} : U_j = 1 \Leftrightarrow \sum_{i \in \mathcal{R}} T_i \times \mathcal{M}_{i,j} \geq 0. \quad (6)$$

Expression (4) states the optimization problem. The set of redundant constraints (5) and (6) permits a stronger filtering during the search.

CP-LNS. We propose here an improved CP model obtaining the same filtering as the original one but resulting in a lighter propagation of the constraints:

$$\text{maximize } \sum_{i \in \mathcal{R}} T_i \times r_i, \quad (7)$$

$$\forall i \in \mathcal{R} : T_i = 1 \Leftrightarrow r_i \geq 0. \quad (8)$$

The objective is to maximize the sum of rows contributions which are formalized as $r_i = \sum_{j \in \mathcal{C}} U_j \times \mathcal{M}_{i,j}$. Each row with positive (respectively negative) contribution is constrained in (8) to be selected (respectively unselected).

This improved model avoids the computation of the quite-heavy reified sum constraints (6) and reduces the number of terms in the objective function. As each product between variables in the objective is translated into a small binary constraint, reducing their number from $|\mathcal{R}| \times |\mathcal{C}|$ to $|\mathcal{R}|$ makes a significant difference on the time spent to compute the fix-point in each node.

CP Global Constraint. We also propose to improve the model due to Le Van et al. [7] by designing a novel algorithm encapsulated inside a global constraint that captures the whole problem. The pseudo-code is given in Algorithm 1. The call to the bounding and filtering procedures has been made explicit. In practice, the lines 11 to 14 would be encapsulated in the global constraint triggered by the fix-point algorithm. The key ingredients of our approach are:

- A feasible solution at each node of the search tree (`evaluate()`).
- An efficient bounding procedure (`upperBound()`).
- An efficient procedure to filter the domains (`filter()`).

A feasible solution at each node of the search tree. CP usually updates its feasible solution and best so far lower bound in the leaf-node of the search tree, that is when every variable of the problem is bound. One can observe that for the max-sum sub-matrix problem, any partial assignment of the variables can be extended implicitly as a complete solution for which the unbound variables would be set to 0 (i.e. $U^?$ variables are considered unselected). There is thus no need to wait that every variable is bound to evaluate the solution and possibly update the best so far lower bound. The value of the objective function of the feasible solution is computed in the `evaluate()` method as the sum of the positive rows contributions of the partial solution (see (2) and (3) where $U^1 = J$): $f(I_{U^1}^*, U^1) = \sum_{i \in \mathcal{R}} \max(0, r_i)$.

An efficient bounding procedure. CP uses a branch and bound depth-first-search to avoid the exploration of proven suboptimal solutions. The branch and bound performances depend on the strength and efficiency of the procedure to compute the upper bound. We design simple yet efficient bounding procedure for the max-sum sub-matrix problem. Intuitively one computes an upper bound on the row contribution of each row and sums up all the positive bounds on the rows. The upper bound on the contribution of a row is the sum of the matrix entries in the selected columns plus the sum of the positive entries in the unbound columns. One simply computes the upper bound as:

$$g(P) = g(U^1, U^0, U^?) = \sum_{i \in \mathcal{R}} \max(0, r_i + \sum_{j \in U^?} \max(0, \mathcal{M}_{i,j})). \quad (9)$$

The bound is admissible but not tight as it may optimistically evaluate the objective ($g(P) \geq f(P)$). Indeed, it relies on a per-line relaxation of the problem, each selecting a possibly different set of columns.

The `upperBound()` method is an implementation of the proposed upper bound using reversible double to store the incremental modifications of the partial contribution of the rows. Using a reversible sparse-set \mathcal{T} for the row variables allows an efficient exclusion or inclusion of the rows through descent or backtrack [17]. Indeed, as soon as the bound on the row contribution becomes negative it should not be considered in the descendant nodes of the search tree. The number of rows to consider goes from exactly $|\mathcal{R}|$ to at most $|\mathcal{R}|$.

An efficient filtering procedure. The `filter()` method evaluates the upper bound result for two one-step look-ahead scenarios: if column j would be selected, this look-ahead upper bound is denoted ub_j^ξ , or if j would not be selected, denoted ub_j^ζ . Then, any column j with $ub_j^\xi \leq best$ is discarded as inclusion of the column can only lead to worst solution than the best so far. Similarly, any column j with $ub_j^\zeta \leq best$ is included. The time complexity for computing all the look-ahead upper bounds is in $O(|\mathcal{T}| \times |U^?|)$. Indeed the look-ahead bound of each line for each column can be obtained in $O(1)$ from r_i^{ub} .

Algorithm 1. CP Global Constraint

```

1  best ← -∞ // best so far objective
2  trail
3  r: array[m] rev-double // rows partial sums
4  T: rev-set ← {1, ..., n} // candidate rows

5  Method dfs()
6  |   if U? ≠ ∅ then
7  |   |   j ← select j ∈ U?
8  |   |   foreach v ∈ {0, 1} do
9  |   |   |   trail.pushState()
10 |   |   |   D(Uj) ← v
11 |   |   |   best ← max(best, evaluate())
12 |   |   |   ub ← upperBound()
13 |   |   |   if ub > best then
14 |   |   |   |   filter()
15 |   |   |   |   dfs()
16 |   |   |   trail.popState()

17 Method evaluate(): double
   // evaluate objective
18   for j ∈ U? do
19   |   if D(Uj) ≠ {0, 1} then
20   |   |   U? ← U? \ j
21   |   |   if D(Uj) = 1 then
22   |   |   |   ri ← ri + Mi,j, ∀i ∈ T
23   |   return ∑i ∈ T max(0, ri)

24 Method upperBound(): double
25   ub ← 0
26   for i ∈ T do
27   |   riub ← ri + ∑j ∈ U? max(0, Mi,j)
28   |   if riub > 0 then ub ← ub + riub
29   |   else T ← T \ i // ri always ≤ 0
30   return ub

31 Method filter(): double
   // remove impossible values
32   ubjξ ← 0 ∀j ∈ U?
33   ubjζ ← 0 ∀j ∈ U?
34   for i ∈ T do
35   |   for j ∈ U? do
36   |   |   if Mi,j > 0 then
37   |   |   |   if riub - Mi,j > 0 then
38   |   |   |   |   ubjξ ← ubjξ + riub - Mi,j
39   |   |   |   ubjξ ← ubjξ + riub
40   |   |   |   else
41   |   |   |   |   ubjζ ← ubjζ + riub
42   |   |   |   |   if riub + Mi,j > 0 then
43   |   |   |   |   |   ubjξ ← ubjξ + riub + Mi,j
44   |   for j ∈ U? do
45   |   |   if ubjξ ≤ best then D(Uj) ← 0
46   |   |   if ubjζ ≤ best then D(Uj) ← 1

```

3.3 Mixed Integer Linear Programming

Mixed Integer Linear Programming [10] involves the optimization of a linear objective function, subject to linear constraints. Some or all of the variables are required to be integer. A MILP solver explores a branch and bound tree using linear-programming (LP) bounds at each node of the search tree.

It differs from a classical branch and bound as a LP *relaxation*, obtained by removing all the integrality constraints of a node, is solved before branching. The domain of all rows and columns variables changes from $\{0, 1\}$ to $[0, 1]$. This relaxed problem can be solved in polynomial time and the solution is an upper

bound on the objective value of the constrained problem. As an upper bound, the LP relaxation solution can be used to prune out suboptimal solutions.

If any integer variable is associated to a fractional value in the LP relaxation, two sub-problems are generated imposing restrictions on the domain of this variable. When all integrality constraints are satisfied in the solution of a node, then it corresponds to a feasible solution and the lower bound is possibly updated.

In an initial formulation, each entry of the matrix is associated to a decision variable that takes the value 1 if and only if both rows and columns are selected. The objective function is computed as the sum of the selected matrix entries.

Initial Model. The max-sum sub-matrix problem can be linearized as:

$$\text{maximize } \sum_{i \in \mathcal{R}, j \in \mathcal{C}} \mathcal{M}_{i,j} \times x_{i,j}, \quad (10)$$

$$\text{s.t. } x_{i,j} \leq T_i, \quad \forall i \in \mathcal{R}, \forall j \in \mathcal{C}, \quad (11)$$

$$x_{i,j} \leq U_j, \quad \forall i \in \mathcal{R}, \forall j \in \mathcal{C}, \quad (12)$$

$$x_{i,j} \geq T_i + U_j - 1, \quad \forall i \in \mathcal{R}, \forall j \in \mathcal{C}. \quad (13)$$

A binary decision variable is associated to each row T_i , to each column U_j and to each matrix entry $x_{i,j}$. The objective function is computed as the sum of matrix entries whose decision variable is set to one. Equations (11) to (13) enforce that variable $x_{i,j} = 1$ if and only if $T_i = 1$ and $U_j = 1$. All these decisions variables are relaxed to the interval $[0, 1]$ in the MILP solver.

Improved Model. In our experiments, reported in Sect. 4, we consider an improved and more compact MILP formulation. It relies on (3) where a row is selected if its contribution is positive and unselected otherwise. For each row, a variable r_i^+ is defined as the contribution of row i if it is positive and 0 otherwise. The objective is to maximize the sum over these variables. This linear model also uses “big M ” constants. More specifically, (16) and (17) linearize $r_i^+ = \max(0, r_i)$ with r_i being the sum of the selected entries of row i .

$$\text{maximize } \sum_{i \in \mathcal{R}} r_i^+, \quad (14)$$

$$\text{s.t. } r_i = \sum_{j \in \mathcal{C}} \mathcal{M}_{i,j} \times U_j, \quad \forall i \in \mathcal{R}, \quad (15)$$

$$r_i^+ \leq T_i \times M_+, \quad \forall i \in \mathcal{R}, \quad (16)$$

$$r_i^+ \leq r_i + (1 - T_i) \times M_-, \quad \forall i \in \mathcal{R}. \quad (17)$$

If $T_i = 1$, $r_i^+ \leq r_i + (1 - T_i) \times M_- \leq T_i \times M_+$, then $r_i^+ \leq r_i$. If $T_i = 0$, $r_i^+ \leq T_i \times M_+ \leq r_i + (1 - T_i) \times M_-$, then $r_i^+ \leq 0$. From the maximization (14), it appears that if $r_i \geq 0$, T_i must be bound to 1 and $r_i^+ = r_i$. Otherwise, T_i must be bound to 0 and $r_i^+ = 0$. This formulation is valid if and only if $M_+ \geq r_i$ and $M_- + r_i \geq 0$. To avoid rounding errors and ill conditioned matrices, the big

M constants can be replaced by $\sum_{j \in \mathcal{C}} \max(0, \mathcal{M}_{i,j})$ and $-\sum_{j \in \mathcal{C}} \min(0, \mathcal{M}_{i,j})$, respectively in (16) and (17).

4 Experiments

This section describes experiments conducted to assess the relative performances of three algorithms to solve the max-sum sub-matrix problem. CP-LNS denotes the improved version of the method CP-LNS₀ proposed by Le Van [7]. The other algorithms are original methods proposed in the present work: a constraint programming with a global constraint (CPGC) and a mixed integer programming (MILP) solution.

These algorithms are first compared on data matrices which are generated in a controlled setting. Experiments on the breast cancer gene expression data used in [7] are reported next. The main criterion to assess the performance of the various methods is the computing time to solve a particular problem instance. This is technically assessed through an any-time profile defined below.

All algorithms have been implemented in the Scala programming language (2.11.4). Each run is executed with a single thread on a MacBook Pro (OS version 10.10.5) laptop (Intel i7-2720 CPU @ 2.20–3.30 GHz, 4 GB RAM per run). Constraint programming implementations are based on the latest version of Oscala [11] and MILP is based on the latest version of Gurobi (7.0.2). The code, datasets and supplementary results are available at <https://bitbucket.org/vbranders/maxsumsubmatriximplementation>.

4.1 Synthetic Data

We follow a similar protocol as in [7]. Synthetic data are generated by implanting a sub-matrix (I, J) of interest in a larger matrix $\mathcal{M} = (\mathcal{R}, \mathcal{C})$ made of m rows and n columns. The implanted sub-matrix (I, J) forms a specific selection of rows and columns chosen at random. For each row index (from 1 to m) and each column index (from 1 to n) of \mathcal{M} , a binary variable is sampled from a Bernoulli distribution $B(p)$ and the associated row or column is included in the sub-matrix (I, J) if $B(p) = 1$. Hence, $I = \{i \in \mathcal{R} \mid B(p) = 1\}$ and $J = \{j \in \mathcal{C} \mid B(p) = 1\}$. Next, the full matrix \mathcal{M} is generated according to two normal distributions, $\mathcal{N}(1, 1)$ whenever the particular entry belongs to the implanted sub-matrix, and $\mathcal{N}(-3, 1)$ otherwise.

Such a generation protocol favors the occurrence of higher values in the implanted sub-matrix and lower values elsewhere. Yet, given the standard deviations chosen equal to 1, both ranges of values may overlap. We note that, as in [7], the implanted sub-matrix is not guaranteed to be an optimal solution to the max-sum sub-matrix problem. This generation protocol looks however realistic to define a rectangular (and not necessarily contiguous) sub-matrix of interest in a larger matrix.

Problem instances are generated for various matrix sizes (m, n) and a varying parameter p . As p increases, the size of the implanted sub-matrix is expected to increase as well. In the gene expression analysis context, m can easily be two orders

of magnitude larger than n and the sub-matrix of interest is typically small as compared to the full matrix. Such cases are included in the controlled experiments reported below but a larger spectrum of problem instances is also considered.

4.2 Gene Expression Data

The proposed case study concerns biomarker discovery for breast cancer subtypes using heterogeneous molecular data types. For a biological analysis and interpretation of the results, the reader is redirected to the work of [7]. The pre-processed data provided by [7] consists of a matrix of $m = 2,211$ rows and $n = 94$ columns.

Matrix entries are first transformed to discrete ranks along each row. A given threshold $\theta \times n$ is then subtracted from each entry. As θ increases, the proportion of positive entries decreases and, consequently, a smaller sub-matrix of interest is expected to be found. Hence, the control parameter θ plays a similar role as the parameter p (from Sect. 4.1) but in an opposite way.

4.3 Evaluation

One could assess algorithms performances through runtime or number of feasible solutions. While the former may depend on implementation details, the latter strongly depends on the time spent in each node. As an example, the large number of reified constraints in CP-LNS₀ has a major impact on the time spent to compute the fix-point in each node while the filtering is as strong as the filtering of the CP-LNS model. While both should perform equally well in terms of the number of feasible solutions, it was observed in preliminary experiments that CP-LNS₀ is significantly slower than CP-LNS (the interested reader may consult <https://bitbucket.org/vbranders/maxsumsubmatriximplementation>). We prefer the runtime comparisons as it is a more common approach and we made sure to implement the algorithms in the most comparable fashion.

Any-Time Profile. In practice, an important criterion for the user is the time required to solve an instance and the ability to find the best solution within a given budget of time. Using *any-time profiles*, one can summarize these characteristics. The idea behind any-time profiles is that an algorithm should produce as high quality solution as possible at any moment of its running time [8]. It directly provides a cumulative probability for a method to solve an arbitrary instance after a given budget of time. In the max-sum sub-matrix problem, a high solution quality corresponds to a sub-matrix of large sum. For each instance, runs not completed in a maximum budget of time t^{\max} are interrupted.

Definition 2 (Max-Sum Sub-Matrix Any-Time Profile). Let $f(\text{algo}, \text{inst}, t)$ be the objective value of the best solution found so far by an algorithm *algo* for an instance *inst* at time t . Let t^{\max} be the maximum running time before

interrupting an algorithm. The any-time profile of an algorithm is the solution quality $Q_{algo}(t)$ computed on all instances as a function of the time:

$$Q_{algo}(t) = \frac{1}{|inst|} \sum_{inst} \frac{f(algo, inst, t)}{f(algo_{inst}^*, inst, t^{\max})}, \quad (18)$$

with $algo_{inst}^* = \underset{algo}{\operatorname{argmax}} f(algo, inst, t^{\max})$.

4.4 Results

Figure 2 presents the any-time profile on 50 synthetic data with 10,000 rows and $p = \{0.05, 0.3, 0.7\}$ for 100 columns (column 1) or 1,000 columns (column 2) and the any-time profile on breast cancer gene expression data with 2,211 rows, 94 columns and variable choices of θ (columns 3 and 4).

Synthetic Data. The CP-LNS method is clearly outperformed by the two other methods. It can barely produce any solution within the allocated time budget. The best approach is CPGC followed by MILP. The reported curves are stopped whenever the proof of optimality is obtained or else the maximal running time is reached. Hence, CPGC also exhibits best results whenever proving optimality is possible in the allocated running time.

Gene Expression Data. Each curve corresponds here to the performance of an algorithm on a single instance, the one obtained for a specific choice of θ . On the whole spectrum of instances considered, the clear winner is CPGC. The most interesting instances are those for which $\theta \geq 0.9$ since such settings correspond to small sub-matrices which are more likely to illustrate an interesting biological pattern. In such cases, the best approaches are CPGC and CP-LNS.

Summary. As expected by the size of the search tree, CP-LNS is sensible to the size of the instance matrix producing barely no results on the larger synthetic instances within the time budget. On the opposite, CPGC achieves the best results. Indeed the model uses a dedicated global constraint with efficient filtering through computation of an upper bound and fast update of the lower bounds. The results of MILP are surprisingly good given its inability to express specialized constraints such as these of CP. This is explained by the benefits of the linear-programming relaxation to tighten the gap between the lower and the upper bounds. The current major issue is related to the “big M ” approach that fails to guide the search in some settings on the gene expression data. When θ is smaller, the “big M ” constant M_- is tighter. As a consequence, the result of the LP relaxation as a higher chance to be a tighter bound. It follows a speed-up of the search as it implies a more efficient pruning of the tree.

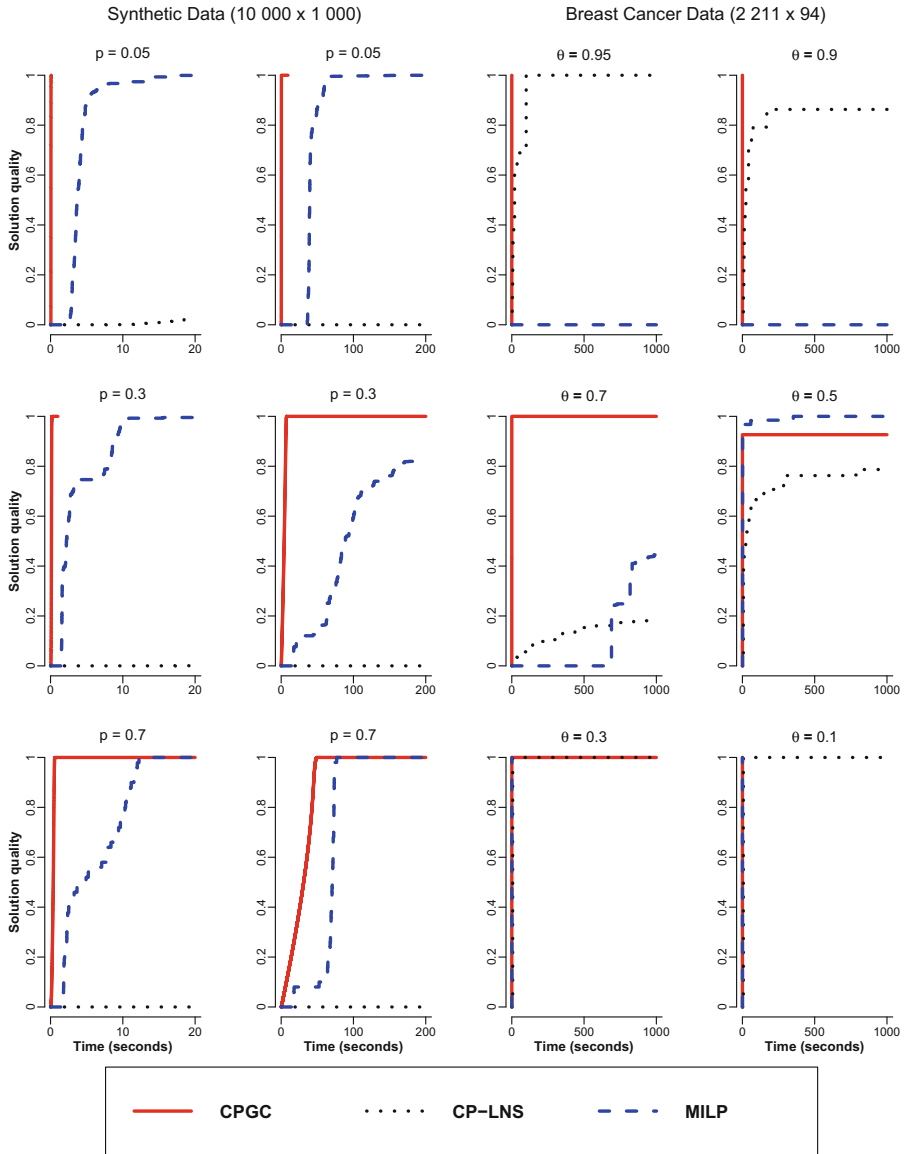


Fig. 2. Any-time profiles of constraint programming with a global constraint (CPGC), the CP method with large neighborhood search (CP-LNS) and mixed integer linear programming (MILP).

Columns 1 and 2: reported curves correspond to the average solution quality over all **synthetic instances** as a function of time (in seconds). Results are computed on 50 **synthetic instances** with 10,000 rows, 100 (column 1) or 1,000 (column 2) columns, a variable p and a maximum running time of 20 (column 1) or 200 (column 2) seconds.

Columns 3 and 4: reported curves correspond to the solution quality over each **gene expression problem instance** obtained for a specific θ as a function of the time (in seconds). Results are computed on breast cancer **gene expression data** with 2,211 rows, 94 columns and various θ values for a maximum CPU time of 1,000 s.

5 Conclusions and Perspectives

We introduce the *max-sum sub-matrix* problem which consists in finding a (non necessarily contiguous) rectangular sub-matrix in a large matrix whose sum is maximal. This problem is originally motivated, in the context of gene expression analysis, by the search of a subset of highly expressed genes in a specific subset, to be found, of relevant samples exhibiting such a pattern. A close variant of this problem, known as *maximal ranked tile* mining problem, has already been studied and tackled with constrained programming (CP) combined with large neighborhood search (LNS) [7].

We present here key properties of the max-sum sub-matrix problem to speed up the search for a solution. This results in an improved CP-LNS implementation. We also propose two new algorithms to solve this problem. Experiments reported both on synthetic data and the original gene expression data used in [7] illustrate the benefits of our proposed methods. In particular, a CP approach with a global constraint (CPGC) is the most effective one in a large spectrum of problem instances. Overall, the CPGC method is also best at proving optimality when such proof can be obtained within the allocated CPU time budget.

The second approach proposed here relies on mixed integer linear programming (MILP). It is arguably the simplest to formulate and to address with a standard solver for such problems. It is competitive with the other methods and largely outperforms CP-LNS as well in our controlled experiments. It exhibits however some performance degradation on some instances from gene expression data, most likely as a consequence of the specific relaxation it is based on.

The max-sum sub-matrix mining problem could be extended to a supervised classification setting. For example, in gene expression analysis, one typically wants to find genes (rows) that allows to discriminate between two conditions. In other words, the columns could be *a priori* labeled according to two conditions. The objective can then be to identify a subset of rows that are maximally relevant to discriminate between subsets of samples from different conditions. This could be encoded in a larger matrix for which columns represent pairs of columns in either conditions from the original matrix and the value stored is interpreted as a distance value for a particular gene across both conditions.




The max-sum sub-matrix problem could also be applied to outlier detection and/or biclustering. For example, using an appropriate data transformation, entries that are close to the mean or to the median could be mapped to relatively large positive entries. Similarly, entries far away from the mean would be mapped to low values. Consequently a sub-matrix of maximal sum after such transformation would correspond to subsets of rows and of columns exhibiting similar entries. Explicit comparisons to existing biclustering algorithms could be considered in such a setting.

References

1. Atzmueller, M.: Subgroup discovery. *Wiley Interdiscipl. Rev. Data Mining Knowl. Discov.* **5**(1), 35–49 (2015)
2. Bentley, J.: Programming pearls: algorithm design techniques. *Commun. ACM* **27**(9), 865–873 (1984)
3. Cheng, Y., Church, G.M.: Biclustering of expression data. In: ISMB, vol. 8, pp. 93–103 (2000)
4. Dawande, M., Keskinocak, P., Tayur, S.: On the biclique problem in bipartite graphs (1996)
5. Fanaee-T, H., Gama, J.: Eigenspace method for spatiotemporal hotspot detection. *Expert Syst.* **32**(3), 454–464 (2015). eXSY-Nov-13-198.R1
6. Herrera, F., Carmona, C.J., González, P., del Jesus, M.J.: An overview on subgroup discovery: foundations and applications. *Knowl. Inf. Syst.* **29**(3), 495–525 (2011)
7. Le Van, T., van Leeuwen, M., Nijssen, S., Fierro, A.C., Marchal, K., De Raedt, L.: Ranked tiling. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) ECML PKDD 2014. LNCS (LNAI), vol. 8725, pp. 98–113. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44851-9_7
8. López-Ibáñez, M., Stützle, T.: Automatically improving the anytime behaviour of optimisation algorithms. *Eur. J. Oper. Res.* **235**(3), 569–582 (2014)
9. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans. Comput. Biol. Bioinform. (TCBB)* **1**(1), 24–45 (2004)
10. Nemhauser, G.L., Wolsey, L.A.: Integer programming and combinatorial optimization. Wiley, Chichester (1988). Nemhauser, G.L., Savelsbergh, M.W.P., Sigismondi, G.S.: Constraint classification for mixed integer programming formulations. *COAL Bull.* **20**, 8–12 (1992)
11. Oscar Team: Oscar: Scala in OR (2012). <https://bitbucket.org/oscarlib/oscar>
12. Parker, J.S., Mullins, M., Cheang, M.C., Leung, S., Voduc, D., Vickery, T., Davies, S., Fauron, C., He, X., Hu, Z., et al.: Supervised risk predictor of breast cancer based on intrinsic subtypes. *J. Clin. Oncol.* **27**(8), 1160–1167 (2009)
13. Perou, C.M., Sørlie, T., Eisen, M.B., van de Rijn, M., Jeffrey, S.S., Rees, C.A., Pollack, J.R., Ross, D.T., Johnsen, H., Akslen, L.A., et al.: Molecular portraits of human breast tumours. *Nature* **406**(6797), 747–752 (2000)
14. Pio, G., Ceci, M., D’Elia, D., Loglisci, C., Malerba, D.: A novel biclustering algorithm for the discovery of meaningful biological correlations between micrnas and their target genes. *BMC Bioinform.* **14**(7), S8 (2013)
15. Pio, G., Ceci, M., Malerba, D., D’Elia, D.: Comirnet: a web-based system for the analysis of mirna-gene regulatory networks. *BMC Bioinform.* **16**(9), S7 (2015)
16. Pontes, B., Giráldez, R., Aguilar-Ruiz, J.S.: Biclustering on expression data: a review. *J. Biomed. Inform.* **57**, 163–180 (2015)
17. de Saint-Marcq, V.I.C., Schaus, P., Solnon, C., Lecoutre, C.: Sparse-sets for domain implementation. In: CP Workshop on Techniques foR Implementing Constraint programming Systems (TRICS), pp. 1–10 (2013)
18. Takaoka, T.: Efficient algorithms for the maximum subarray problem by distance matrix multiplication. *Electron. Not. Theoret. Comput. Sci.* **61**, 191–200 (2002)
19. Tamaki, H., Tokuyama, T.: Algorithms for the maximum subarray problem based on matrix multiplication. In: SODA 1998, pp. 446–452 (1998)
20. Yang, J., Wang, H., Wang, W., Yu, P.: Enhanced biclustering on expression data. In: Proceedings of the Third IEEE Symposium on Bioinformatics and Bioengineering, pp. 321–327. IEEE (2003)



A Scaled-Correlation Based Approach for Defining and Analyzing Functional Networks

Samuel Dolean¹ , Mihaela Dînsoreanu¹ , Raul Cristian Mureşan² ,
Attila Geiszt¹, Rodica Potolea¹, and Ioana Ţincas²

¹ Department of Computer Science, Technical University of Cluj-Napoca,
Cluj-Napoca, Romania

samm.dlln@yahoo.co.uk, {Mihaela.Dinsoreanu,

Rodica.Potolea}@cs.utcluj.ro, attila.geiszt@gmail.com

² Transylvanian Institute of Neuroscience, Cluj-Napoca, Romania

{muresan,tincas}@tins.ro

Abstract. Many natural systems can be described as networks of interacting elements, forming a graph of interactions. This is the case for climate models, coupled chemical systems, computer or social networks, or the brain. For many of these cases, dynamical networks emerge whose structure changes in time. Estimating the structure of such networks from the time series that describe the activity of their nodes is a serious challenge. Here, we devise a new method that is based on the Scaled Correlation function to estimate interactions between nodes that occur on fast timescales. We apply the method on EEG measurements from human volunteers to evaluate neuronal functional connectivity associated with a visual perception task. We compare the statistics of networks extracted with the new method with those that are extracted using traditional techniques, like the Pearson correlation coefficient or the cross-correlation function. Results indicate that the new method is superior in identifying networks whose structure correlates to the cognitive processes engaged during visual perception. The method is general enough to be applied on any data that describes dynamical interactions evolving on multiple timescales, as is the case in climate modeling, chemical networks, or complex biological systems.

Keywords: EEG · Functional brain networks · Metrics
Cross-Correlation · Scaled-Correlation · Pearson correlation coefficient
Directed weighted network

1 Introduction

Network science [1] has become a major research field in the past decade, relying heavily on computational methods to extract and characterize networks from complex data. A wide variety of systems can be described as connected graphs,

from computer networks, social interaction patterns, disease spreading models, to biological networks. Networks can even be found in the legal system [2] and tourism [3]. Importantly, many of these networks are dynamical, in the sense that individual nodes exhibit a time-dependent activity that shapes interaction patterns with the other nodes. Such is the case of complex brain networks [4], whereby different brain regions connect and disconnect all the time as a function of cognitive processing, thus yielding functional networks whose structure evolves in time. Similarly, in climate modeling one constructs networks from modeled space-time series [5], yielding complex dynamical graphs. A large effort has been dedicated to characterizing complex networks but less attention has been paid to how these networks are defined and extracted from the data, although this is a critical step. For dynamical networks, each node has an associated activity that can be described as a time series. To evaluate if two nodes are functionally connected, the traditional procedure is to use the pairwise Pearson correlation coefficient [6] or the cross-correlation function [7]. Arguably, both of these methods are sub optimal because they either do not consider the full temporal structure of the data (e.g., delays), or cannot distinguish faster from slower interactions, respectively. Here we develop a novel method to extract networks from complex time series using the Scaled Correlation function (SCF) that can estimate interactions on the fast timescales by means of restricted sampling [8]. We apply this method on a hard test case, defining and characterizing functional brain networks from high-density EEG signals recorded in humans during a demanding visual task. The challenge is to determine if statistics of networks extracted using SCF correlate better to the cognitive task that participants have to solve than those extracted with traditional techniques. To this end, we focused on three different types of networks: Pearson Coefficient Weighted Networks (PCWN), Cross Correlation Weighted Networks (CCWN), and Scaled Correlation Weighted Networks (SCWN) and for each we evaluated network theory metrics.

2 Related Work

Three types of neural connectivity are considered in the literature [9]: structural, functional, and effective. Structural connectivity pertains to the physical, anatomical connections between brain areas and is considered to be fixed on a short term. By contrast, functional connectivity expresses sub-graphs of the anatomical network that are transiently coupled, depending on the activity of the nodes [10]. Effective connectivity constrains the graph further by considering only those interactions that mediate the reciprocal influence of brain areas [9].

Here we focused on functional connectivity, as it is more widely used and easier to estimate. The basic theoretical framework for graph/network analysis of functional connectivity is given in [4], with more advanced metrics being defined in [11]. The community and hub structure dynamic was studied in [12], that concluding that increasing the cognitive task difficulty leads to lower modularity, fewer provincial hubs, and more connector hubs. The relevance of the network size was studied in [13], showing that different metrics depend on it (i.e. clustering coefficient, modularity, efficiency, economic efficiency and assortativity).

The conclusion was that efficiency, assortativity were higher and modularity was lower on large networks compared to smaller networks, even though their density was the same.

Functional brain connectivity is usually estimated by computing pairwise correlations between activities of different neural populations. The most popular measure of correlation is the Pearson Correlation coefficient (PCC). Also, a related approach was presented in [14] by computing partial correlations between pairs of signals. Partial correlation consists of calculating PCC but augmenting this coefficient in order to eliminate the influence of potential third-party signals. One of the most important conclusions is that using first grade partial correlations the distribution of values are centered around zero whereas non-partial correlations (simple PCC) were spread along the $[0, 1]$ interval.

PCC as well as its partial counterpart ignore the multiple timescales present in neural signals. For example, fast oscillations in the gamma band (30–80 Hz) are expressed in relation to a plethora of cognitive and perceptual processes but traditional measures, like the PCC, cannot selectively evaluate the fast-timescale correlations induced by such oscillations. By contrast, we have developed a measure called Scaled Correlation [8], which isolates correlations expressed on fast timescales by using restricted sampling.

Here, we analyzed EEG data recorded from human volunteers performing a visual recognition task. Participants had to identify objects from images of stimuli containing deformed grids/lattices of dots (see the “Dots” method for a reference). The varying deformation of these lattices made recognition easier or harder. Our objective was to evaluate how functional brain connectivity changes when the subjects engage in the perceptual task, compared to the “baseline” condition (i.e., before the stimulus was shown on the screen). We wanted to determine which measure of connectivity is able to more efficiently reveal the reorganization of functional networks during perceptual engagement.

3 Relevant Concepts

A **participant** is defined as one of the individuals that took part in the experiment, and for which specific data was recorded.

A **trial** is a part of an experiment, time-wise. An experiment is divided into several trials (in our case, 210 trials), and each trial contains some events. In this experiment the trials have different lengths, as the participants were free to explore. An **event** consists of a specific time instant relative to the beginning of the trial and a unique code which has significance for the experiment.

A **correlogram** is the result of a correlation function [8] applied on two signals. As the correlogram is an array of values, the **peak** of a correlogram is the maximum absolute value along all values. The **lag** is the position where the **peak** was found in the correlogram.

The **stimulus** refers to the moment when the picture is displayed on screen. We call **baseline** the moment right before the stimulus appears.

An **area** is one of the brain regions (occipital, frontal, parietal, left temporal and right temporal). In our case it is represented by a group of electrodes (whose coordinates are given by the headset used in experiment).

4 Identification of Functional Networks

For each participant and for each trial, we do the following steps: first, using the recorded signals, we construct graphs using different correlations to estimate the functional connectivity, and then we apply metrics from complex network theory on these graphs in order to investigate the relevance of the metrics and to analyze properties of the network (Fig. 1). These steps are further explained in the following paragraphs.

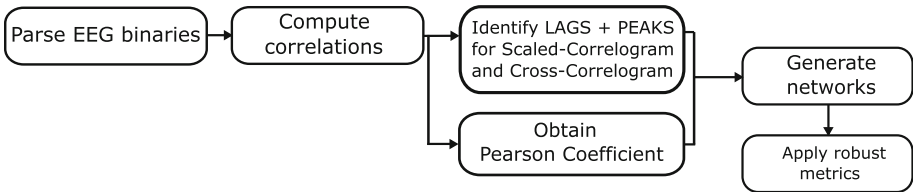


Fig. 1. Lags, Peaks and Pearson weighted networks approach used

The first phase consists of parsing the EEG signals recorded from the participants. Based on these signals we compute various correlation functions. We use three types of functions: Cross-Correlation (CCF), Scaled-Correlation (SCF) [8] and Pearson Correlation Coefficient (PCC). Based on the resulted cross/scaled-correlogram (in case of CCF and SCF) we identify lags and peaks. In the case of PCC we simply use it without further processing. The decision of using PCC is based on its frequent use in literature. However this coefficient fails to represent the timing relation (i.e., delay) between signals, its value representing only the instantaneous correlation at lag zero. In contrast, SCF and CCF slides the signals in time enabling to identify delays that indicate potential causality relations between signals.

In the next phase, we create five kinds of graphs, referred by us as the PCCWN (Peaks Cross Correlation weighted Network), LCCWN (Lags Cross Correlation weighted Network), PSCWN (Peaks Scaled Correlation weighted Network), LSCWN (Lags Scaled Correlation weighted Network) and the PCWN (Pearson Coefficient weighted Network). All these graphs have 128 nodes, corresponding to the 128 EEG channels.

The *PCWN* is an undirected weighted graph which has the absolute values of the Pearson correlation coefficient (samples version) as weights between any two nodes. The Pearson correlation coefficient shows how linearly correlated two signals are varying from: 1 - the signals are perfectly linearly correlated, to -1 - they are perfectly anti-correlated.

The *PCCWN* and *PSCWN* are directed graphs having edges with weight defined as the absolute value of the largest peak from the cross/scaled-correlogram [8], respectively. The *LSCWN* and *LCCWN* are directed graphs as well but the weights of the edges are the absolute values of the lags of the peaks identified previously. The absolute peak value is the strongest correlation value at time t . As we mentioned a negative correlation indicates the anti-correlation which is perfectly fine because it means that most probably the two signals come from opposite electrode sites. The lag of this peak indicates the delay of information transfer hence the bigger the distance from 0, the higher delay it is obtained no matter if it is negative or positive. In the case of the *LSCWN*, *LCCWN*, *PSCWN*, and *PCCWN* the direction of the edge is defined by the lag sign. Considering channels A and B, a positive lag shows B leading A while a negative lag shows A leading B. When the lag is 0, we consider that the channels are instantaneously correlated and we assign two bidirectional edges (from A to B and from B to A). For channels with the same index (diagonal) the value in the matrix is zero.

For the window of the CCF we have chosen a window of ± 100 ms, which has the effect of focusing on small delays. For the SCF we applied a scale window segment of 25 ms to keep only correlations between components that have a frequency greater than 40 Hz. We have chosen to use Pearson-based and Scaled-Correlation based networks to compare results obtained with the traditional method with those which extract only the fast correlations of the signals, considered to be important for conscious visual perception [15]. However we decided to go for Cross-Correlation as well because we want to understand if considering the temporal structure (delays) of signals brings additional benefits compared to the PCC. The CCF is used without any normalization.

For each network we decided to reduce its density (keeping only 50% of the edges with the strongest weights). The motivation behind the thresholding is that we also considered the reduction of potential noises that may alter the correlation values [16]. Another reason for density reduction is because of the extremely high density of PCWN which is a complete graph, hence the metrics would lead to improper results. Furthermore, we chose to take the absolute value for edge weights because negative edge weights are affecting graph metrics (i.e., Average Path Length) while the absolute value still captures the information about the correlation strength.

For the second analysis (per areas) we didn't consider a 50% density reduction since we are interested in raw values of the peaks and lags as they were initially computed. In order to be consistent with the previous strategy for the metrics, we kept the absolute values too in this new analysis.

The following measures have been applied: average path length (APL) [17, 18], global clustering coefficient (GCC) [19], betweenness centrality (BC) and closeness centrality (CC) [17]. For each measure we study the modifications of the global network statistics by comparing CC, SC and PCC.

APL has been considered on the obtained graphs and the computation was done by using Dijkstra shortest path algorithm [17]. In order to achieve the

average path length based on the shortest paths between each two nodes, the following formula has been applied:

$$apl_{G_{weighted}} = \frac{1}{N(N-1)} \sum p_{i,j}, i \neq j \quad (1)$$

where $p_{i,j}$ is the shortest path between node i and j and N is the total number of nodes. Because this metric is a distance based metric we applied it only on the lags network. However in the case of the Pearson network, we consider stronger connections as closer connections by inverting the edge weights in the computation of the shortest path.

5 Experimental Results

5.1 Data Description

Electroencephalography (EEG) data was recorded from 10 healthy human volunteers performing a visual recognition task. A high-density Biosemi ActiveTwo machine, with 128 channels was used to record scalp potentials with a sampling rate of 1024 samples/s. The experimental protocol followed the one described in [20], with several important modifications. Participants were shown visual stimuli that represented shapes of 30 objects through a lattice of dots that was distorted to capture object contours (see the ‘‘Dots method’’ in [20]). The experiment was organized in 7 successive blocks of 30 stimuli, each block being characterized by a different level of distortion. The first block contained no distortion of the lattice, and thereby no information about the object, whereas the seventh block contained the maximal distortion, enabling effortless recognition of the objects. Blocks were shown in ascending order of distortion, thus rendering objects increasingly more visible in successive blocks. This yielded a total of 210 trials (30 stimuli/block x 7 blocks). Compared to our previous study [20], here we used only a subset of 30 stimuli that were validated in a pilot experiment – only objects that provided unambiguous recognition were included. Also, data was recorded from 10 novel participants with an ‘‘ascending’’ protocol only, i.e. increasing visibility in successive blocks. Data from all the 10 subjects was analyzed.

Each individual trial consisted of several periods that were delineated by using TTL pulses on an 8-bit line delivered to the EEG machine by a National Instruments PCI-6503 board controlled by the stimulation computer (see Fig. 2). Trial start was signaled by an event code (trigger) 128 and was followed by the presentation of a red fixation dot in the center of the screen, which subjects had to watch for 1–1.5 s. A new trigger value of 150 was then followed by a white, full screen mask kept for 500 ms. Then, a 129 trigger code was issued simultaneously with the presentation of the dot stimulus on screen. Subjects were free to visually explore the stimulus and then had to press one of three keys, signaling that they (i) have seen the object and can name it (trigger code = 1), or (ii) have seen something but are uncertain about the object it represents (trigger code = 2), or (iii) haven’t seen any object (trigger code = 3). Each baseline (segment between triggers 150 and 129) and stimulus (segment between trigger 129 and trigger 1, 2, or 3) periods of a trial

in the raw EEG data was represented as a matrix of floating-point values, where each row corresponded to a channel (128 rows) and columns corresponded to the samples during the baseline and stimulus period, respectively (see Fig. 2). Different participants had a different number of trials for each response type: e.g., the first participant had 63 seen trials, 53 uncertain trials and 94 trials where he recognized nothing. Other participants had different numbers of trials for each response type. Once all trials are parsed and gathered from the raw data for each response category, we computed the CCF, SCF and PCC for each pair of channels. The networks we obtained were represented as square (128×128) adjacency matrices, where we ignored the primary diagonal (self-connectivity of nodes). Matrices for PCCWN (based on CCF) and PSCWN (based on SCF) contained, for each row-column pair a value between $[-1, 1]$ representing the value of the largest peak in the correlogram. In the case of LCCWN and LSCWN, this value was taken to be the lag where the peak was positioned in the correlogram. Finally, for PCWN (Pearson correlation coefficient) the value was between $[-1, 1]$ (no information about lag was available).

To avoid having all-to-all connectivity, for PCCWN, LCCWN, PSCWN, LSCWN and PCWN the density was reduced by eliminating 50% of the weakest edges. In case of lag networks (LCCWN and LSCWN), the density reduction was done by considering the corresponding correlation value of the peak in the correlogram (PCCWN/PSCWN) instead of the lag.

On the extracted networks, we applied two different types of network metrics: distance based (APL, BC, CC) and connection based (GCC). In order to allow a comparison between PCCWN, PSCWN and PCWN (peak based networks) we created a set of binary networks by keeping only the strongest 50% of the links. Because we end up with a binary network for the PCWN (if the value is zero, then it is a zero otherwise it is a one), binary networks are considered as well for the PCCWN and PSCWN in order to allow results comparison. In the next section we describe the comparisons between Cross Correlation and Scaled Correlation based networks and Pearson Correlation based networks. In addition to exploring the full 128 node networks, we also grouped the nodes according to the anatomical position of their corresponding electrodes. We used this strategy in order to estimate the interaction between brain areas (occipital, parietal, frontal, temporal) during the visual task. We applied this grouping method only for CCF and SCF based networks (as will be shown, the Pearson network did not offer informative results). Results for each individual area were labeled with a capital letter (e.g. Occipital - O, Frontal - F, Left Temporal - LT etc.).

5.2 Results

After generating the candidate functional brain networks, we applied the metrics mentioned previously in order to identify how these metrics change from baseline to stimulus periods and as a function of the type of connectivity measure that was used to define the networks.

Figure 3 (top row) displays the average of APL across the 10 subjects, as a function of the perceptual condition (seen, uncertain, unseen) and depending

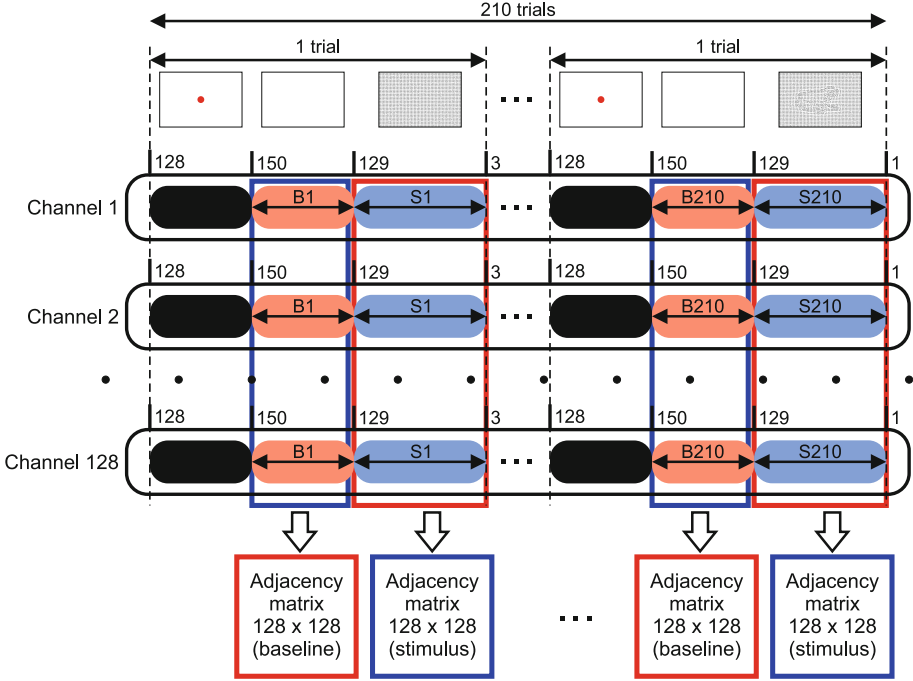


Fig. 2. Schematic representation depicting how the data was extracted from the raw signals. For each participant, 128 signals were recorded as successive trials were presented. A red dot was presented on screen (black segment), followed by a full screen white mask (red segment), after which the stimulus was shown on the screen (blue segment) until the subject pressed a key to signal that (i) the stimulus was recognized (code = 1), (ii) the subject was uncertain about it (code = 2), (iii) the subject saw nothing relevant (code = 3). A 128×128 matrix with pairwise correlation values or lags was extracted for each segment corresponding to baseline (B1, B2, ..., B210) and stimulus presentation (S1, S2, ..., S210), yielding a total of 210 matrices associated to baseline and 210 matrices associated to stimulus periods for each subject. (Color figure online)

on the type of network. Clearly, the largest difference between the baseline and stimulus periods is exhibited by LSCWN, indicating a strong reduction in APL when the brain is engaged in perceptual processing. By contrast, neither LCCWN nor PCWN networks showed such a consistent and strong effect.

In Fig. 3 (second row) we show results for GCC, whereby this measure was consistently lower for all networks during the baseline than during the stimulus period. Notably, the largest increase in GCC induced by stimulus presentation was again exhibited by PSCWN, with a close result for PCCWN. Again, PCWN performed the worst in terms of showing differences between baseline and stimulus periods.

The BC and CC are illustrated in Fig. 3 (third and fourth rows). As was the case with the other metrics, BC and CC were modulated the strongest for

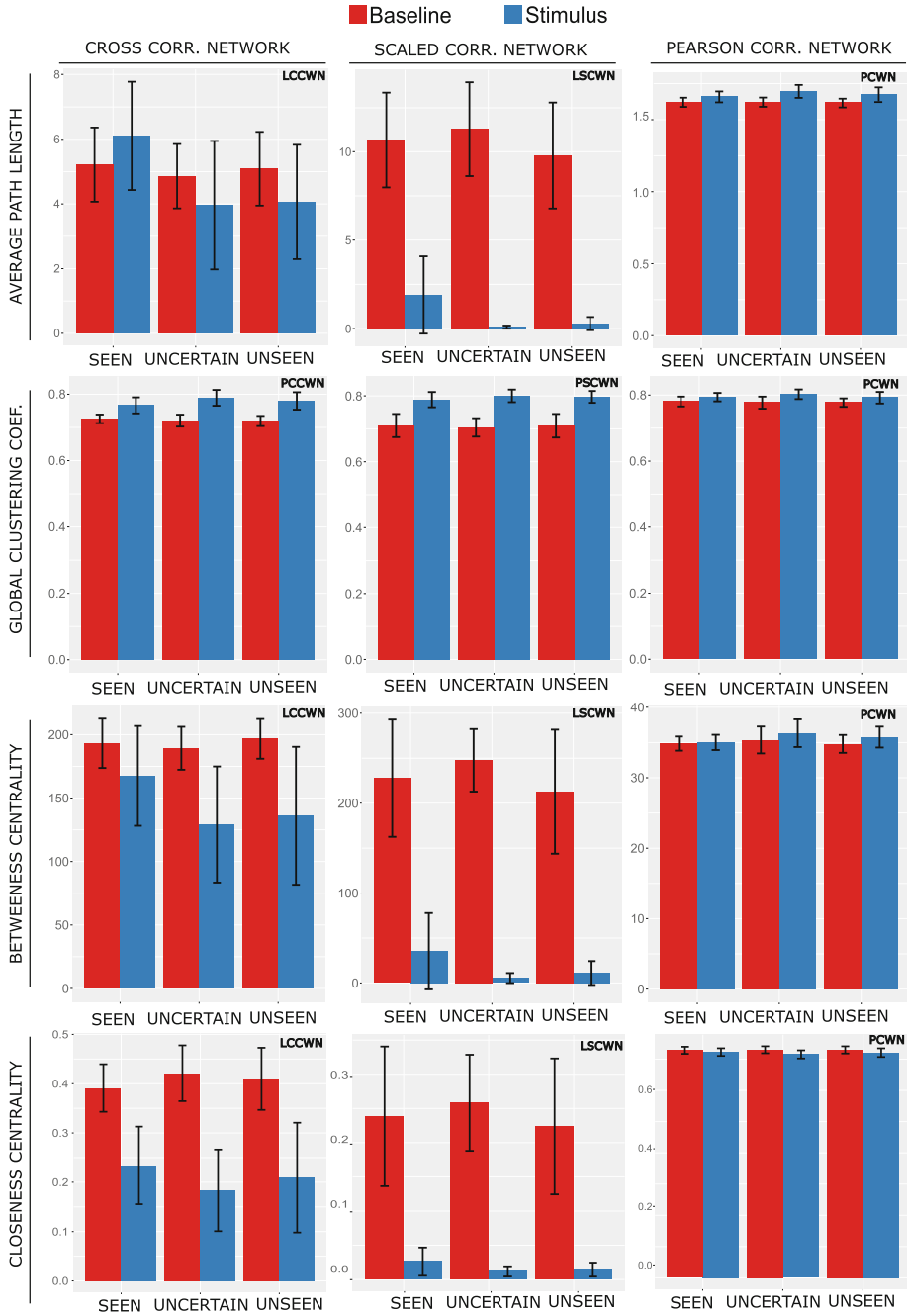


Fig. 3. Metrics applied on our generated networks. APL, BC and CC applied on LCCWN, LSCWN and PCWN. GCC applied on PCCWN, PSCWN and PCWN. For the Scaled Correlation the segment size $s = 25$ ms (fast events - 40Hz). For all networks the density was reduced with 50%. The value obtained is the mean value across all trials and then across all 10 subjects. Error bars are S.D.

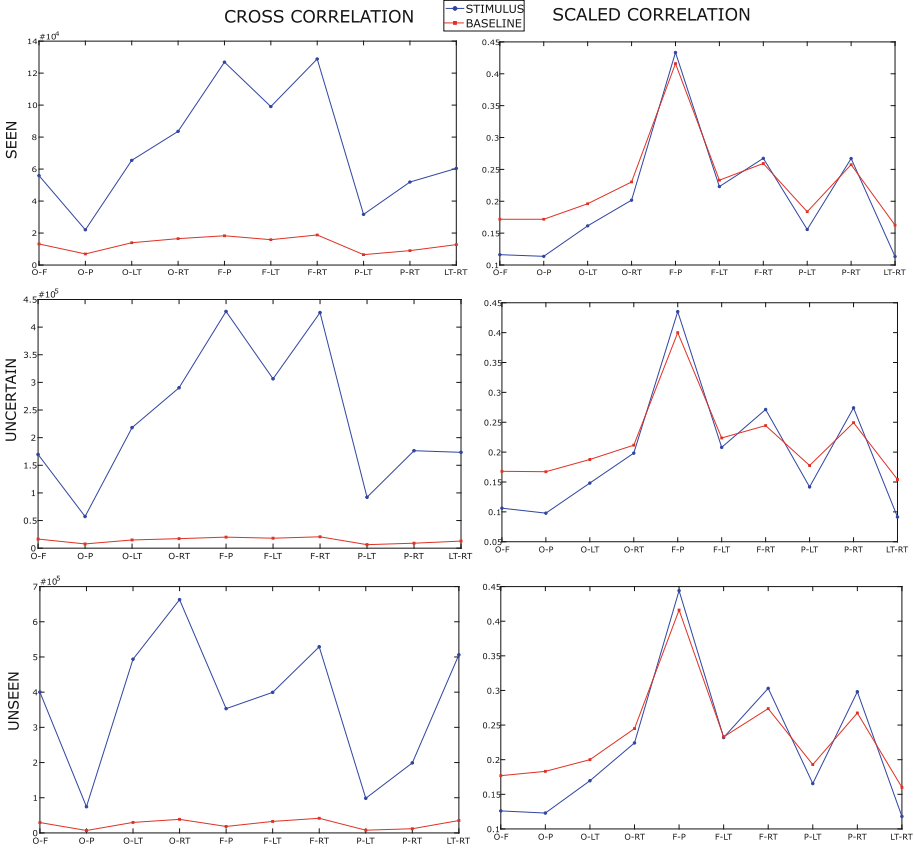


Fig. 4. Areas pairs for PCCWN and PSCWN

the case of networks extracted with SCF and somewhat less for networks whose definition relied on CCF. Importantly, PCC-based networks did not show a clear modulation of any of the two metrics.

Area Level Analysis. Since SCF and CCF based networks showed the largest difference between baseline and stimulus, we next focused only on the PCCWN and PSCWN networks. We computed the average across all link weights (average absolute peak correlations). We expected to see higher correlations for the O-F pair when the stimulus was on the screen and a lower correlation in the baseline, because the two areas are actively involved in conscious visual processing. As can be seen in Fig. 4 the PCCWN (Cross Correlation) presents the expected behavior: the correlation between Occipital and Frontal is higher when the stimulus is on the screen meaning that these two areas are better correlated by exchanging more information during the stimulus than during the baseline period. Also was a similarity across all perceptual outcomes (seen, uncertain and unseen) which may indicate that the O-F correlation is non-specific,

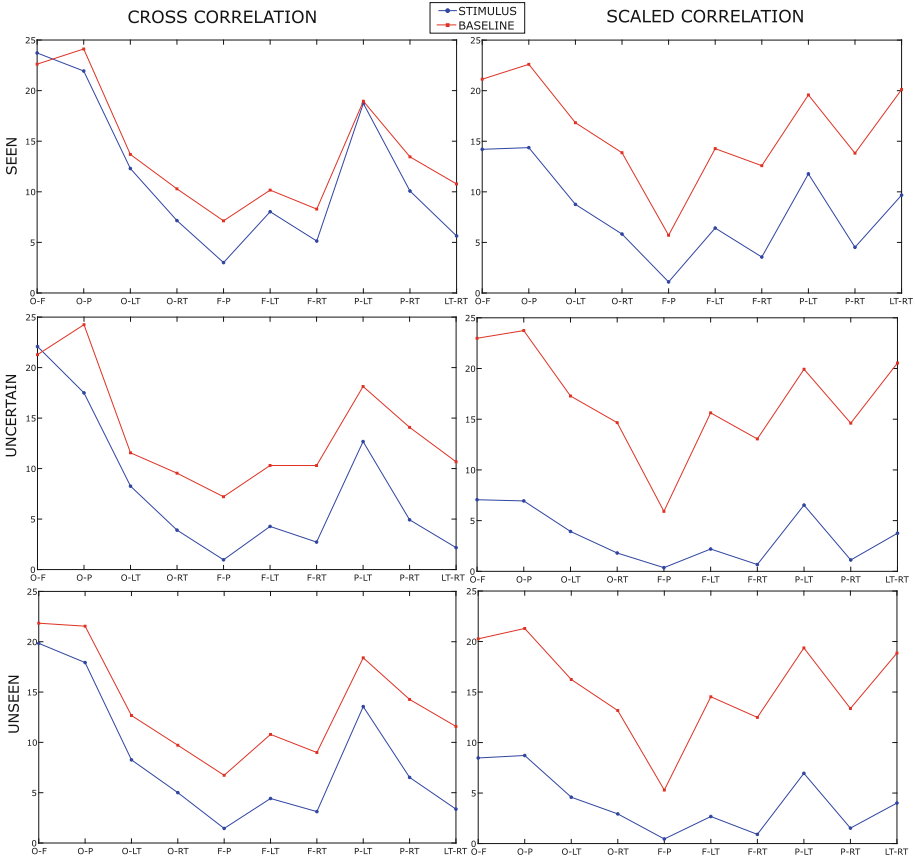


Fig. 5. Areas pairs for LCCWN and LSCWN

i.e. it reflects visual processing but not the decision taken by the subject. The effects in the PSCWN (Scaled Correlation) were smaller and this behavior may be the result of considering absolute values of the correlation. For the lags network we expect to see the opposite behavior: a lower lag when the stimulus is shown and a higher lag in the baseline. Figure 5 shows the mean lags for each pair. The most important thing to notice is the consistency across all perceptual outcomes (seen, uncertain, unseen). Another behavior that can be observed for PCSWN (Scaled Correlation) is the lag being lower (closer to 0) for the stimulus period. This indicates the fact that the overall information may be moving faster during cognitive engagement.

6 Conclusion

We have shown that SCF enables the extraction of functional networks from complex time series in a way that outperforms traditional ones because it can isolate

those networks that evolve on particular timescales. As a result, the structure of SCF-extracted networks more readily correlates with the cognitive processes that support visual perception in humans. This is likely due to the fact that fast, gamma oscillations (30–80 Hz) are known to correlate to visual perception, in particular [21], and cognitive processing, in general [22]. SCF is able to estimate the fast interactions between neural populations that occur in the gamma range and reveal the dynamical networks evolving on the fast timescale.

Results also show that PCC-based networks performed the worst in distinguishing the baseline from the stimulus periods. CCF-based networks fared better, indicating that information about the temporal relation of signals (delays) is highly relevant in order to properly define the functional brain networks associated to stimulus processing. On top of this temporal information, SCF brings the further advantage that it can select those networks that evolve on the fast timescales, which are known to be co-expressed with perceptual and cognitive processes. The present study suggests that SCF networks may be the preferred choice as they can help define networks that are more likely reflecting the relevant underlying functional brain networks.

The method we have introduced is useful for the investigation of complex brain networks. For example, it could help identify how dynamics of functional networks are altered in brain disorders [23] and may find interesting applications for brain-computer interfaces [24]. However, the applicability of the method is not restricted to neuroscience problems. The ability of SCF to extract fast networks from time series opens its applicability range to a wide array of issues where dynamical networks can be found. For example, in climate modeling thermal disturbances or humidity evolve on various timescales and particular networks may be identified for each. Another example pertains to modeling of coupled chemical processes whose reaction rates may also cover a range of timescales. For all these cases, SCF can enable the selective investigation of dynamical networks that evolve on different timescales. To conclude, the method we have introduced proves very useful for studying brain networks but it is general enough to lend itself to the analysis of dynamical networks from a wide array of research areas.

Acknowledgement. This work was supported by two grants from Consiliul National al Cercetării Științifice (CNCS) - Unitatea Executivă pentru Finanțarea Învățământului Superior, a Cercetării Dezvoltării și Inovării (UEFISCDI): PNII-RU-TE-2014-4- 13 0406/2015 contract no. 169/2015 and PN-III-P4-ID-PCE-2016-0010 contract no. 78/2017.

References

1. Barabasi, A.L.: *Network Science*. Cambridge University Press, Cambridge (2016)
2. Koniaris, M., Anagnostopoulos, I., Vassiliou, Y.: *Network analysis in the legal domain: a complex model for European Union legal sources*, CoRR (2015)
3. Baggio, R., Scott, N., Cooper, C.: *Network science: a review focused on tourism*. *Ann. Tour. Res.* **37**, 802–827 (2010)
4. Bullmore, E., Sporns, O.: *Complex brain networks: graph theoretical analysis of structural and functional systems*. *Nat. Rev. Neurosci.* **10**(4), 312 (2009)

5. Lange, S., Donges, J.F., Volkholz, J., Kurths, J.: Local difference measures between complex networks for dynamical system model evaluation. *PLoS ONE* **10**(4), e0129413 (2015)
6. Pearson, K.: Notes on regression and inheritance in the case of two parents. *Proc. Royal Soc. Lond.* **58**, 240–242 (1895)
7. Bracewell, R.: Pentagram notation for cross correlation. In: *The Fourier Transform and Its Applications*, pp. 46 and 243. McGraw-Hill, New York (1965)
8. Nikolić, D., Mureşan, R.C., Feng, W., Singer, W.: Scaled correlation analysis: a better way to compute a cross-correlogram. *Eur. J. Neurosci.* **35**(5), 742–762 (2012)
9. Friston, K.J.: Functional and effective connectivity: a review. *Brain Connect.* **1**(1), 13–36 (2011)
10. Park, H.J., Friston, K.: Structural and functional brain networks: from connections to cognition. *Science* **342**(6158), 1238411 (2013)
11. Rubinov, M., Sporns, O.: Complex network measures of brain connectivity: uses and interpretations. *NeuroImage* **52**(3), 1059–1069 (2010). *Computational Models of the Brain*
12. Finc, K., Bonna, K., Lewandowska, M., Wolak, T., Nikadon, J., Dreszer, J., Duch, W., Khn, S.: Transition of the functional brain network related to increasing cognitive demands. *Hum. Brain Mapp.* **38**, 3659–3674 (2017)
13. Joudaki, A., Salehi, N., Jalili, M., Knyazeva, M.G.: EEG based functional brain networks: does the network size matter? *PLoS ONE* **7**(4), 1–9 (2012)
14. Jalili, M., Knyazeva, M.G.: Constructing brain functional networks from EEG: partial and unpartial correlations. *J. Integr. Neurosci.* **10**(2), 213–232 (2011)
15. Meador, K.J., Ray, P.G.: Gamma frequency coherence and conscious perception. *J. Clin. Neurophysiol.* **16**(2), 170 (1999)
16. Bordier, C., Nicolini, C., Bifone, A.: Graph analysis and modularity of brain functional connectivity networks: searching for the optimal threshold. *Front. Neurosci.* **11**, 441 (2017)
17. Opsahl, T., Agneessens, F., Skvoretz, J.: Node centrality in weighted networks: generalizing degree and shortest paths. *Soc. Netw.* **32**(3), 245–251 (2010)
18. Reppas, A.I., Spiliotis, K., Siettos, C.I.: Tuning the average path length of complex networks and its influence to the emergent dynamics of the majority-rule model. *Math. Comput. Simul.* **109**, 186–196 (2015)
19. Opsahl, T., Panzarasa, P.: Clustering in weighted networks. *Soci. Netw.* **31**(2), 155–163 (2009)
20. Moca, V.V., Țincaș, I., Melloni, L., Mureşan, R.C.: Visual exploration and object recognition by lattice deformation. *PLoS ONE* **6**(7), e22831 (2011)
21. Singer, W.: Neuronal synchrony: a versatile code for the definition of relations? *Neuron* **24**(1), 49–65 (1999)
22. Fries, P., Nikolić, D., Singer, W.: The gamma cycle. *Trends Neurosci.* **30**(7), 309–316 (2007)
23. Stam, C.J.: Modern network science of neurological disorders. *Nat. Rev. Neurosci.* **15**, 683–695 (2014)
24. Jarosiewicz, B., Chase, S.M., Fraser, G.W., Velliste, M., Kass, R.E., Schwartz, A.B.: Functional network reorganization during learning in a brain-computer interface paradigm. *Proc. Natl. Acad. Sci. U.S.A.* **105**(49), 19486–19491 (2008)



Complex Localization in the Multiple Instance Learning Context

Dan-Ovidiu Graur, Răzvan-Alexandru Mariş^(✉), Rodica Potolea,
Mihaela Dinşoreanu, and Camelia Lemnaru

Technical University of Cluj-Napoca, Cluj-Napoca, Romania
{dan.graur,razvan.maris}@student.utcluj.ro

Abstract. This paper introduces two approaches for solving Multiple Instance Problems (MIP) in which the traditional instance localization assumption is not met. We introduce a technique which transforms individual feature values in the attempt to align the data to the MIP localization assumption and a new MIP learning algorithm which identifies a region enclosing the majority (negative) class while excluding at least one instance from each positive (minority class) bag. The proposed methods are evaluated on synthetic datasets, as well as on a real-world manufacturing defect identification dataset. The real-world dataset poses additional challenges: data with noise, large imbalance and overlap.

Keywords: Multiple instance learning
Axis-Parallel Hyper-Rectangle · Feature value transformation
R-APR · Classification

1 Introduction

The aim of this work is to design a systematic strategy to detect faults in industrially manufactured entities. The real-world dataset originates from the traceability system of a *Printed Circuit Board* production line, therefore its dimension is considerably large (≈ 320 GB). Each entity is composed of a variable number of components. The characteristics of every component are known. After being manufactured, entities are labeled as functional or faulty by automatic inspection machines. An entity may be faulty due to one or more components. The task is to define a model that is able to identify non-functional entities. The difficulty arises due to the fact that faulty entities can contain both non-functional and functional components, without them being explicitly differentiated. This is known in literature as the *Multiple Instance Problem (MIP)*. In the current context, one expects the components rendering the entity to which they belong faulty (*positive instances*) to have atypical characteristics compared to functional components (*negative instances*). However, classical MIP algorithms attempt to find regularities amongst positive instances. In other words, MIP algorithms expect positive instances to be located in a small, dense region, while negative instances are supposed to be scattered around the feature space. In the studied problem,

however, positive instances are situated in a large, less dense region (possibly scattered), while negative instances are located in a denser region. Although *outlier* or *novelty* detection techniques could be used under these circumstances, the region defining positive instances is not necessarily well determined. As such, our objective is to propose a novel set of methods through which such atypical MIP problems can be solved.

2 The Multiple Instance Problem

The MIP comes as a generalization to the classical Supervised Learning Problem [3], in that training examples consist of groups of instances, where an instance is a feature vector. Each such group is known as a *bag*, and each such bag has an associated label. That is to say, labels are not directly associated to an instance, but rather to a group of instances. The concept which stands at the foundation of the MIP is known as *the standard MIP assumption* [12], or *linearity hypothesis* [5], which states that a *positive* bag has at least one positive instance, whilst a *negative* bag has no positive instances. The MIP is considerably more difficult than classical Supervised Learning [6, 7], mainly due to the high degree of noise introduced in the learning process by the arbitrarily high number of positive instances a positive bag can have [7]. As such, specialized MIP algorithms need to be employed, to tackle the problem at hand.

Following, is a formal description of the standard MIP, based on the notation in [13]. Let $B = \{B_1, B_2, \dots, B_m\}$ be a *set of m bags*, where $\forall B_i \in B \exists v_i \in \mathbb{N}^*$, such that $B_i = \{B_{i1}, B_{i2}, \dots, B_{iv_i}\}$ is a bag containing v_i n -dimensional feature vectors. Let B_{ij} be the j^{th} instance of the i^{th} bag, such that $B_{ij} = \{B_{ij1}, B_{ij2}, \dots, B_{ijn}\}$. Let $L = \{l_1, l_2, \dots, l_m\}$ be the label set, and $l_i \in Y$ for $i = 1 \dots m$. In the particular case of *binary classification*, which is the problem approached in this paper, $Y = \{\perp, \top\}$. Finally, let $D = \{\langle B_1, l_1 \rangle, \langle B_2, l_2 \rangle, \dots, \langle B_m, l_m \rangle\}$ be the labeled data. The aforementioned standard MIP assumption can be formally represented as $l_i = l_{i1} \vee l_{i2} \vee \dots \vee l_{iv_i}$, that is, a bag is positive if and only if it has at least one positive instance.

Whilst the *standard MIP* is arguably the most popular type of MIP, it is important to mention that the MIP context hosts a set of more complex challenges [1, 2, 12]. Weidmann et al. [12] produce a comprehensive taxonomy of the various types of MIPs, based on the existence of a multitude of underlying concepts, as opposed to a singular underlying concept which stands at the foundation of the *positive* class, as is the case in the standard MIP. These challenges are not to be further detailed here, since they are beyond the scope of this paper.

3 MIP Issues in the Current Context

The MIP de facto standard works under the assumption that positive instances converge towards a certain region, whilst negative instances are scattered around the feature space. However, in the given context, positive instances are scattered around the feature space, while negative instances cluster within a particular

region. The *Antisymmetry Problem* (AP) is best described graphically by Fig. 1. It might be tempting to consider that a simple class label inversion solves this problem. However, this is not the case, since positive bags will now consist only of positive instances, while negative bags will contain both negative and positive instances. This goes against the assumptions made by existing MIP algorithms, and as a consequence, their learning process becomes biased. For instance, the Iterated Discrimination [6] algorithm requires only one instance from every positive bag to be included in the resulting Axis-Parallel Hyper-Rectangle (APR). However, after the label inversion, every instance belonging to a now positive bag is positive. Therefore, this algorithm would yield a high number of false negatives (or false positives, considering the initial labels). Another issue is the presence of positive instances in negative bags, as mentioned previously, which may prove problematic during the *feature selection* stage. Likewise, the DD metric [7], which stands at the foundation of the EM-DD algorithm [13], will require extensive modification in order to accommodate the existence of negative bag instances in high density areas. Thus, existing methods require either a preprocessing step or changes in their approach to allow them to tackle the AP.

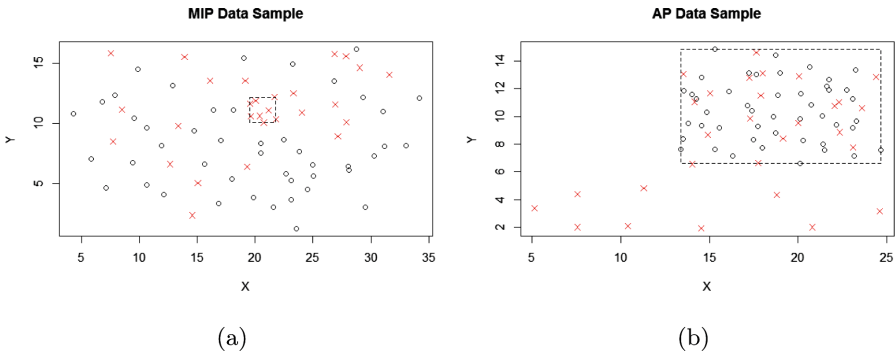


Fig. 1. × = positive bag instance and o = negative bag instance. (a) Positive instances converge. (b) Positive instances are scattered.

3.1 A Feature-Value Transformation Based Approach

Our approach first transforms the feature space to meet the MIP instance localization assumption. Such a transformation is supposed to bring positive instances “closer” together while scattering negative instances around the feature space. Such a transformation would apply a function $f : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$, where n is the number of features, to all instances \mathbf{x} , replacing their feature vectors with $f(\bar{\mathbf{x}}, \mathbf{x})$, where $\bar{\mathbf{x}}$ is the mean of all instances belonging to negative bags:

$$\bar{\mathbf{x}} = \frac{\sum_{\mathbf{x} \in N} \mathbf{x}}{|N|}, \text{ where } N \text{ is the set of negative instances} \tag{1}$$

The function f must be chosen such that positive instances end up “closer” to $\bar{\mathbf{x}}$, while negative instances end up “further” from $\bar{\mathbf{x}}$, considering a metric for which a difference in only one dimension of the feature vectors is enough for the output to change considerably (e.g. the *Euclidean* metric). It must be noted that this specific transformation relies upon the fact that all negative instances are clustered. The dataset obtained after applying the transformation is then fed into the Iterated Discrimination algorithm [6].

It is worth mentioning, however, that the feature-value transformation employed here is *general purpose*, and as such, can be used on any antisymmetric dataset, whose initial structure is incompatible with the standard MIP algorithms, to convert it so that MIP learning methods can be applied.

3.2 An Axis-Parallel Hyper-Rectangle Based Approach

The second approach we propose towards solving the standard MIP in the Antisymmetric MIP Context is the *Reverse Axis-Parallel Hyper-Rectangle Algorithm* (R-APR). R-APR is inspired by the *Iterated Discrimination algorithm* [6]. It solves the standard MIP by finding an APR which, unlike the one resulted from the *Iterated Discrimination algorithm*, encloses all the negative bag instances and some of the instances of positive bags, leaving at least one positive bag instance outside. As such, a bag is classified as positive, if at least one of its instances falls outside of the APR along at least one dimension, whilst a bag is classified as negative if all its instances fall within the APR’s bounds for all dimensions.

The algorithm consists of four major stages: All-Negative APR Generation, High Density Positive Instance Margin Expansion, Feature Selection, and finally, Statistical Margin Expansion. The R-APR algorithm attempts to solve the AP without employing any sort of feature-value transformations, other than normalization. Moreover, the APR produced by this algorithm yields valuable information in terms of what the normal value ranges for the relevant features are. Consequently, in certain contexts, such as that of industrial manufacturing, it provides potentially useful insight into the production process.

4 Solving the Antisymmetry Problem

This section provides a more in depth description of the two original approaches we propose towards solving the AP problem in the MIP.

4.1 The Transformation-Based Iterated Discrimination Algorithm

This approach requires the definition of a function as described in Sect. 3.1. Every instance is then replaced with $f(\bar{\mathbf{x}}, \mathbf{x})$, with the purpose of bringing positive instances “closer” to $\bar{\mathbf{x}}$ while moving negative instances “further” from $\bar{\mathbf{x}}$. An example of such a function f is:

$$f(\bar{\mathbf{x}}, \mathbf{x}) = \bar{\mathbf{x}} + \frac{\mathbf{x} - \bar{\mathbf{x}}}{\|\mathbf{x} - \bar{\mathbf{x}}\|} \cdot g(\|\mathbf{x} - \bar{\mathbf{x}}\|), \quad (2)$$

where $\|\cdot\|$ is the *Euclidean* norm and $g : \mathbb{R} \rightarrow \mathbb{R}$ is a monotonically decreasing function. The function g can be defined independently of the number of features of the dataset, but then $\|\mathbf{x} - \bar{\mathbf{x}}\|$ must be scaled accordingly. Therefore $g(\|\mathbf{x} - \bar{\mathbf{x}}\|)$ from Eq. (2) should be replaced with $g\left(\frac{\|\mathbf{x} - \bar{\mathbf{x}}\|}{\sqrt{n}}\right)$. This is because the *Euclidean metric* of an n -dimensional vector, whose components are all equal to a , is $\sqrt{n} \cdot a$. That is, $\|(a, a, \dots, a)\| = \sqrt{n} \cdot a$. The *Euclidean* norm is used so that one feature value being “far” from that feature’s mean suffices for the instance to be brought “closer” to $\bar{\mathbf{x}}$.

Figure 3 contains plots of one family of functions which meet the above requirements, described by:

$$\mathbb{G} = \{g : \mathbb{R} \rightarrow \mathbb{R} \mid g(x) = c \cdot a^{-b \cdot x}\}, \text{ where } a, b, c \in \mathbb{R}_{>0}. \quad (3)$$

An exponential family of functions was chosen because the absolute value of their derivative can be made large enough so as to achieve a substantial separation margin between positive and negative instances, regardless of the initial value of this margin. Furthermore, the behavior of these functions in the proximity of 0 can be constrained. The *fixed points* (x_0, y_0) of these functions are marked at the intersection of the vertical line $x = x_0$ with the functions’ plots. Considering Eq. (2), these fixed points and the value of $\|\mathbf{x} - \bar{\mathbf{x}}\|$ determine whether \mathbf{x} ends up closer or further from $\bar{\mathbf{x}}$.

Figure 2 shows the effect of applying (2) to a normally distributed two-dimensional dataset. The function g is replaced in (2), in turn, by the functions displayed in Fig. 3.

4.2 The R-APR Algorithm

The R-APR algorithm consists of the four steps shown in Fig. 4, excluding the data normalization stage, which is optional. The four steps are presented in the subsections that follow.

All-Negative APR Generation. This APR defines a region in feature space which encloses all negative instances. The upper margins of the APR, along every relevant feature d , are defined as:

$$ub_d = \max_{B_i \in B^-, B_{ij} \in B_i} (B_{ijd}) \quad (4)$$

Respectively, the lower bounds are obtained using:

$$lb_d = \min_{B_i \in B^-, B_{ij} \in B_i} (B_{ijd}) \quad (5)$$

Due to the standard MI assumption, the generated APR is not yet ready to be used for classification, since positive bags still have negative instances, which may be outside the All-Negative APR. During this stage, only negative bags are processed.

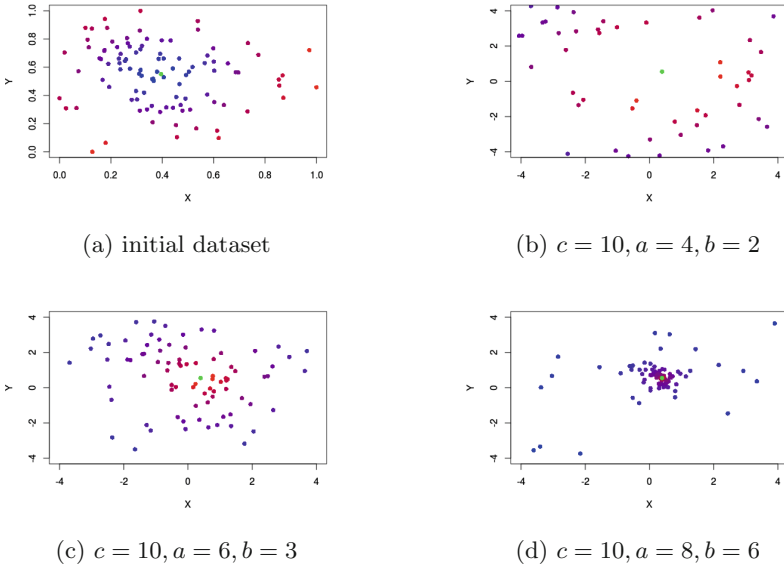


Fig. 2. Figure 2a represents a normally distributed, two-dimensional dataset. Instances that are “closer” to the *mean point* (green) are colored in *blue*, while instances that are “further” from it are colored in *red*. Figures 2b, c and d illustrate the original dataset transformed using Eq. (2), where the function g belongs to the function family described in Eq. (3). (Color figure online)

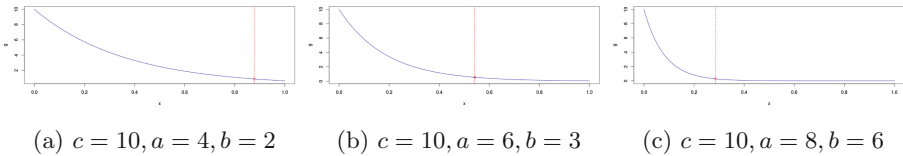


Fig. 3. Plots of functions belonging to the family defined in (3). The vertical lines mark the fixed points of the functions. In this context, the fixed point discriminates between instances x which end up “closer” and “further” from \bar{x} .

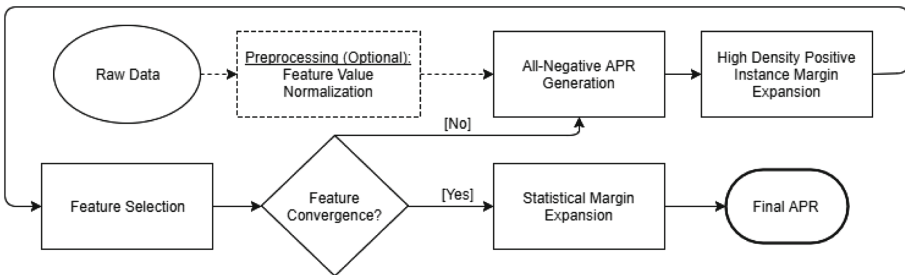


Fig. 4. The general execution flow of the R-APR algorithm.

High Density Positive Instance Margin Expansion. To generalize better, the APR must be expanded in such a way as to include negative instances belonging to positive bags. However, due to the asymmetry [3,7,8] introduced by the bag level label, identifying them is not straightforward. We propose two solutions towards solving this problem, based on the assumption that negative instances from positive bags are gathered together, since they should have similar feature values. Both procedures are based on density and distance measurements.

The first approach refers to selecting one instance from each positive bag, thus constructing a set of instances which are used towards building an auxiliary APR. The new APR is used to expand the All-Negative APR, where necessary. The instance is chosen based on a Density measurement, which computes the instance’s degree of proximity to the other instances belonging to the same bag:

$$HD_i = \arg \max_{B_{ij} \in B_i} \left(\sum_{k, k \neq j} \frac{1}{\zeta + \|B_{ij} - B_{ik}\|^2} \right) \quad (6)$$

where $\|\cdot\|$ is the Euclidean Metric, $\zeta \in \mathbb{R}$ is an offset, and HD_i is the highest density instance of B_i . Additionally, the instance’s distance from the All-Negative APR, is computed, using the Manhattan Distance:

$$dist_{ij} = \sum_d f(B_{ijd}, lb_d, ub_d) \quad (7)$$

where function f is computed as:

$$f(x, lb, ub) = \begin{cases} lb - x, & \text{if } x < lb \\ x - ub, & \text{if } x > ub \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

During this stage, the algorithm attempts to find regions towards which it expands the margins of the All-Negative APR, by essentially *speculating* which regions host a large number of negative instances belonging to positive bags. As such, one must ensure that the APR is not wrongly expanded towards regions of positive instances, as may be the case when positive bags have few negative instances. In order to avoid such cases, the algorithm requires two user-defined thresholds, *density* and *distance*, empirically identified and tuned for every data set. An instance is only selected if its density is above the density threshold, and if its distance from the APR is below the distance threshold.

The second approach we propose comes as an extension to the previously described technique. After determining the highest density instance for a bag, using Eq. (6), an optimization algorithm is used to find the point which maximizes the density function. The search is bounded by a rectangular region defined by the bag’s instances. Once more, the user-defined *distance* and *density* thresholds are used when selecting the instances. The set obtained as a result of the selection procedure is used to construct a new APR, with the aim of expanding the All-Negative APR’s bounds, where necessary.

Feature Selection. Similarly to the Iterated Discrimination algorithm [6], the R-APR algorithm attempts to select the relevant features in an *iterative* fashion. However, unlike the Iterated Discrimination algorithm, discrimination is performed on the positive bags and at the bag-level.

There are two criteria for establishing when a feature discriminates a bag, both dependent on a user-specified global out-of-bounds threshold $t \in \mathbb{R}_{\geq 0}$. Since discrimination is performed at the bag level, a bag’s out-of-bounds value for a particular feature d is given by $val_d = \max_{B_{ij} \in B_i} (out_of_bounds(B_{ij,d}, lb_d, ub_d))$. The *first criterion* specifies that a feature d discriminates a bag B_i if $val_d > t$. The *second criterion* specifies that a feature d discriminates a bag B_i if $val_d > val_k \forall k \in \mathcal{F}_r, d \neq k$, where \mathcal{F}_r is the relevant feature set. Figure 5 describes these concepts visually.

Following is a formal description of the Feature Selection stage: let $\mathcal{F}_r^{old} = \{f_1, f_2, \dots, f_n\}$ be the old set of relevant features. Let $\mathcal{F}_r^{new} = \emptyset$ be the new set of relevant features, initially empty, and let $B_{FS}^+ = B^+$ be the set of positive bags used in the current feature selection stage. As previously mentioned, the Feature Selection stage is iterative. Let f'_i be the most discriminating feature, i.e. the feature which discriminates the most bags in B_{FS}^+ , as identified in iteration i of this stage. Let $B_{f'_i}^+$ be the set of positive bags discriminated by f'_i . It follows that $\mathcal{F}_r^{old} = \mathcal{F}_r^{old} \setminus \{f'_i\}$, and $\mathcal{F}_r^{new} = \mathcal{F}_r^{new} \cup \{f'_i\}$. Moreover, $B_{FS}^+ = B_{FS}^+ \setminus B_{f'_i}^+$. This stage will continue to loop, until either $B_{FS}^+ = \emptyset$ or $\mathcal{F}_r^{old} = \emptyset$. In the former case, the algorithm loops back to **All-Negative APR Generation**, with \mathcal{F}_r^{new} as the set of relevant features. In the latter case, the feature set converges, and the algorithm moves on to the next, and final stage.

Statistical Margin Expansion. The final stage of the R-APR algorithm refers to expanding the margins of the APR obtained so far, to generalize better. It is

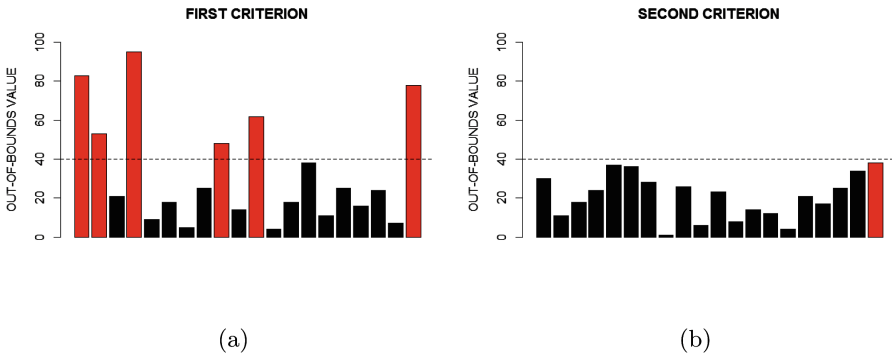


Fig. 5. An example of the two Feature Selection criteria. Each chart contains 20 features (vertical bars), the threshold t is set to 40. Bars represent a bag’s maximal out-of-bounds value along that feature. Red bars represent features that discriminate the bag. Black bars represent features which do not discriminate the bag. (a) Example of the first criterion. (b) Example of the second criterion. (Color figure online)

identical to that employed in the Iterated Discrimination algorithm [6], however, the *Kernel Density Estimation* (KDE) is built from negative bag instances, as opposed to positive bag instances. This stage is controlled by two user-defined constants: ε and τ . The τ constant specifies the amount of probability which should fall within the bounds of the APR, based on the KDE, if the negative bag instances were centered between its bounds. The value of τ is used to establish the deviation σ_d along relevant feature d , such that a normal distribution centered in $\mu_d = \frac{lb_d + ub_d}{2}$ with deviation σ_d hosts τ probability between the upper and lower bounds. This can be formally expressed as $\Pr(lb_d < X < ub_d) = \tau$.

A Gaussian KDE is built for each relevant feature d . Next, relative to the obtained KDEs, the margins of the APR along every relevant feature are expanded so as to ensure that $\frac{\varepsilon}{2}$ probability remains above the upper bound, and that $\frac{\varepsilon}{2}$ remains below the lower bound.

This step is, however, optional, and can be removed when the APR should be tighter. Such cases are largely identified empirically, and are typically due to a high number of positive instances being located near the bounds, outside of the unexpanded APR. Through expansion, these instances are included in the APR, which likely leads to an undesirably high rate of false-negatives.

Classification. Classification assumes a series of comparisons against the bounds of the APR. As such, for an unseen bag, if at least one instance falls outside the APR, along at least one dimension, then that bag is classified as positive. Otherwise, the bag is classified as negative.

The most computationally intensive step in the R-APR algorithm is the **High Density Positive Instance Margin Expansion**. If we consider that the number of instances in a positive bag is constant - which, in the current application context is satisfied (since the number of instances is the number of pads on a component), then the complexity of the R-APR algorithm is linear in the number of positive instances in the entire dataset.

5 Experimental Evaluation

We have evaluated the R-APR algorithm and the effect of the feature-value transformation on the APR algorithm for both synthetic and real-world datasets, using a 5-fold cross validation strategy. The synthetic datasets have 100 positive bags and 100 negative bags, each individual instance having 3 features. Negative bags contain 10 to 19 negative instances. Positive bags contain 10 to 16 negative instances with a probability of 0.5 and 3 to 5 positive instances. Negative instances belong to a 3-dimensional *Gaussian* distribution with $\mu = [10 \ 10 \ 10]$, with no feature correlation, each having a standard deviation $\sigma = 4$. For the positive instances we have employed the same strategy, $\mu = [\alpha_1 \ 10 \ \alpha_2]$, with $(\alpha_1, \alpha_2) \in M_{low} = \{(22, -1), (25, 22), (27, 0), (-3, 25), (-2.5, -2.5)\}$ and $\sigma \in D = \{4, 4.5, 5\}$. In this setting, the degree of overlap is relatively low. We have also considered a dataset with a larger degree of overlap, by using, for the positive instances, $(\alpha_1, \alpha_2) \in M_{moderate} = \{(20, 1), (0.5, 21), (21.5, 0.5), (22, 22), (1, 1)\}$,

with the same values for σ as before. Each positive instance is generated from a tuple chosen randomly and independently from $\{(\alpha_1, 10, \alpha_2) \mid (\alpha_1, \alpha_2) \in M_{low}\} \times D$, in the case of the dataset having a relatively low degree of overlap, and $\{(\alpha_1, 10, \alpha_2) \mid (\alpha_1, \alpha_2) \in M_{moderate}\} \times D$, in the case of the dataset with a larger degree of overlap.

For the feature-value transformation, we have varied a between 6 and 9 and b between 2 and 6, both with an increment of 1. For R-APR, we have varied the *distance* and *density* thresholds for the density based expansion step: *distance* between 0.06 and 0.24, with a 0.02 increment and *density* between 1.3 and 1.8, with a 0.05 increment. For the R-APR tests, we have fixed the parameters of the Statistical Margin Expansion step to $\tau = 0.98$ and $\epsilon = 0.02$.

The average recall and precision values obtained by the APR algorithm with different parameter settings for the feature transformation, on the two different versions of artificial datasets (low and moderate overlap), are presented in Fig. 6.

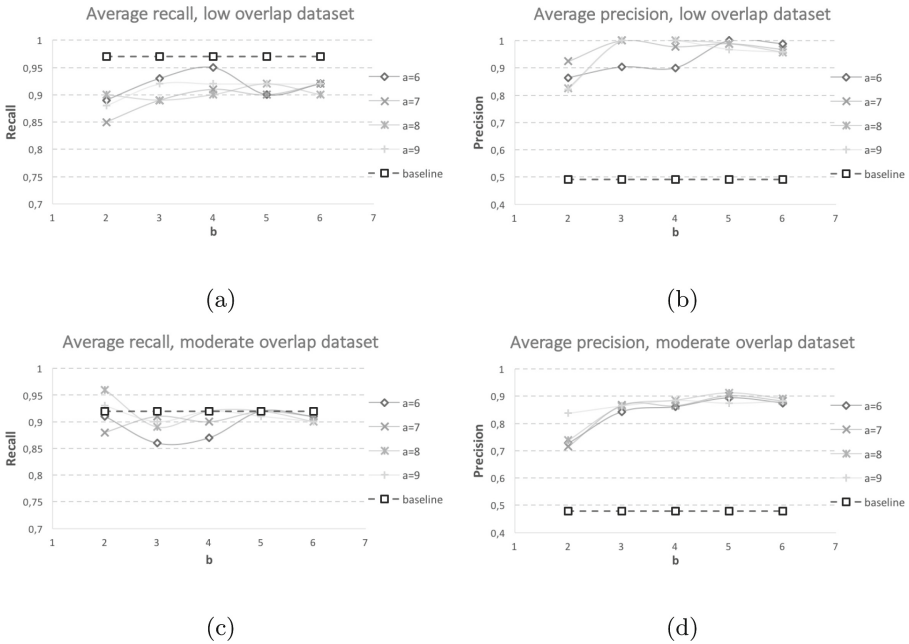


Fig. 6. Average Recall and Precision obtained by the APR algorithm with different transformation settings, the low (a, b) and moderate overlap (c, d) synthetic datasets.

We have included, as baseline, the results obtained by the basic APR algorithm (without the transformation). The feature transformation significantly boosts the precision values obtained by the APR algorithm, on both types of datasets, while keeping recall at approximately the same level. We can observe that as the overlap increases, the recall is maintained at the same level as the

one obtained by the APR algorithm alone, but precision almost doubles. Larger values for a and b produce slightly better precision values. The best trade-off between the precision and recall (considering equal costs for false positives and false negatives) is around the values 7 and 8 for a , and 4 and 5 for b .

The diagram in Fig. 7 shows how recall varies with the different threshold values considered for *distance* and *density*, for the R-APR algorithm, on the two synthetic datasets. We observe that, as we increase the *distance* and decrease the *density* threshold, recall decreases - which is expected, since the risk of expanding the R-APR too aggressively increases with larger *distance* and smaller *density* thresholds. The precision value for these experiments was 1, regardless of the variations considered, whereas the precision obtained by the APR algorithm is 0.49 on the low overlap dataset and 0.48 on the medium overlap dataset (see Fig. 6). However, on the moderate overlap dataset, the boost in precision comes at the high cost of a decay in recall, due to the aggressive expansion of the R-APR in the density based expansion stage. We believe lower values for the *distance* threshold and larger values for the *density* threshold could produce better results in this case.

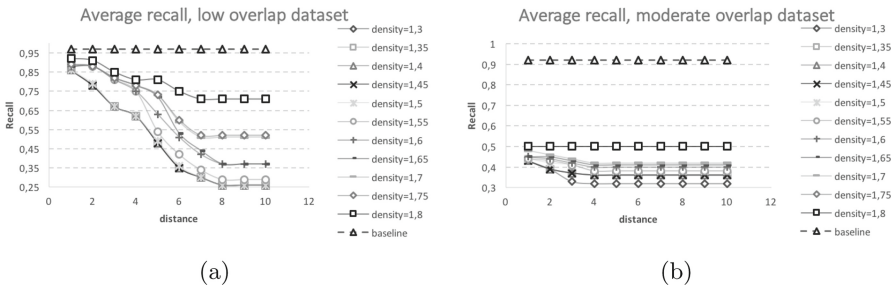


Fig. 7. Average Recall obtained by the R-APR algorithm, with different values for *distance* and *density* thresholds, on the low (a) and moderate overlap (b) synthetic datasets.

The real-world defect identification dataset represents electronic components having multiple pins, the label being associated with the entire component instead of the individual pins. For example, an integrated circuit may consist of around 100 pins and may be labeled as faulty or non-faulty. The dataset on which the following evaluation was performed contains 100 positive bags (faulty components) and 700 negative bags (non-faulty components). Bags contain between 16 and 200 instances. It suffers from severe overlap given the current set of features, therefore the R-APR algorithm yields low recall values. The results obtained by the APR with and without transformation (*baseline*) are presented in Fig. 8. Here again we considered two different values for a (7 and 8) and three values for b (2, 3 and 4). As observed on the synthetic datasets, the transformation can improve precision without degrading recall (for $a = 7$, $b = 3$, recall decreases only slightly, but precision is improved by 0.1 – from 0.86 to 0.96).

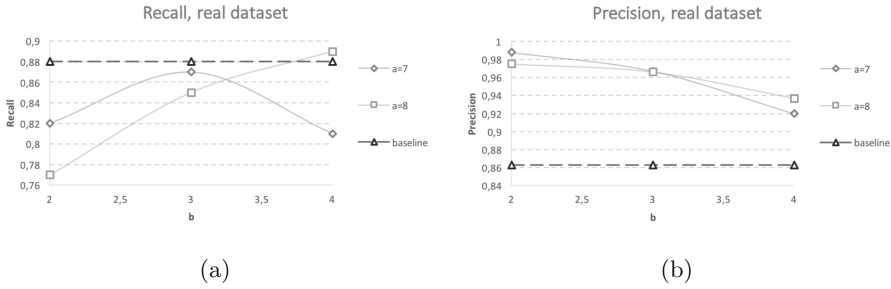


Fig. 8. Average Recall and Precision obtained by the APR algorithm with the transformation, on the real-world dataset.

6 Related Work

The Iterated Discrimination algorithm [6] is one of the traditional MIP algorithms. It attempts to find an APR, such that any bag having an instance within its bounds, is classified as *positive*. Otherwise, the bag is considered *negative*. The method is based on the idea that positive instances have similar features, thus, converging towards a particular region in feature space.

The EM-DD algorithm [13] is another method employed towards solving the MIP. The algorithm is based on the same supposition that positive instances converge towards a particular region. Consequently, it searches for a target point in feature space around which the positive instances are assumed to gather. A bag is classified as positive if at least one of its instances neighbors the aforementioned target. The algorithm is based on the Diverse Density metric [7], which yields high values for hypothesized points in regions containing a large number of instances from diverse positive bags, and low values for regions containing instances from negative bags, or little diversity in terms of positive bags.

Numerous other approaches have been employed towards solving the standard MIP, including: Neural Networks [8], Support Vector Machines [3, 4], density based approaches [7], Lazy Learning [11], Decision Trees, or Rule Sets [5]. Solutions to the standard MIP have also been explored in the context of real-valued labels through methods such as Multiple Instance Regression [10]. MIP algorithms can be applied in many areas, including: image classification, stock prediction, biochemistry, or text classification. However, empirical studies suggest that no particular MIP algorithm appears to perform successfully in every possible problem domain [9]. That is, MIP algorithms vary in performance, depending on the problem they attempt to solve. It is worth emphasizing that unlike the R-APR algorithm, these existing methods are unsuitable for directly solving the AP, without prior feature-value transformation.

7 Conclusions

The paper presents two strategies for tackling the MIP in an antisymmetric complex context, where the positive instances are located in a larger, less dense

area, whilst the negative instances converge towards a particular region. The first technique applies a feature-value transformation in order to align the data with the assumption considered by standard MIP algorithms. The second technique refers to a novel algorithm, the Reverse Axis-Parallel Hyper-Rectangle (R-APR) algorithm, designed to identify a region in feature space, which encloses all the negative bags, while excluding at least one instance from every positive bag. The strategies achieved a good performance on synthetic data: the transformation significantly boosts the precision of the APR algorithm, while maintaining recall at high levels; the R-APR algorithm can achieve a precision of 1, but as the overlap in the data increases this gain comes at the cost of reduced recall values. We believe this deficiency can be partially minimized via the *distance* and *density* threshold values. The real-world data we employed in our experiments suffers from severe overlap, which makes the classes partly indistinguishable given the current set of features. This prevents the R-APR algorithm from achieving satisfactory recall levels; the feature-value transformation, however, manages to boost the precision of the APR algorithm while keeping recall at the same levels, which is also of interest in this particular context, since reducing false positives can lead to reduced manual verification costs. Improving recall remains the primary objective in this scenario and we are currently focusing on identifying new features which could be extracted from the production line in order to reduce overlap and improve the classification outcome.

References

1. Alpaydin, E., Cheplygina, V., Loog, M., Tax, D.M.: Single- vs. multiple-instance classification. *Pattern Recogn.* **48**(9), 2831–2838 (2015)
2. Amores, J.: Multiple instance classification: review, taxonomy and comparative study. *Artif. Intell.* **201**, 81–105 (2013)
3. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: *Proceedings of the 15th International Conference on Neural Information Processing Systems, NIPS 2002*, pp. 577–584. MIT Press, Cambridge (2002)
4. Bunescu, R.C., Mooney, R.J.: Multiple instance learning for sparse positive bags. In: *Proceedings of the 24th International Conference on Machine Learning, ICML 2007*, pp. 105–112. ACM, New York (2007)
5. Chevaleyre, Y., Zucker, J.-D.: Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. Application to the mutagenesis problem. In: Stroulia, E., Matwin, S. (eds.) *AI 2001. LNCS (LNAI)*, vol. 2056, pp. 204–214. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45153-6_20
6. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.* **89**(1–2), 31–71 (1997)
7. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10, NIPS 1997*, pp. 570–576. MIT Press, Cambridge (1998)
8. Ramon, J., Raedt, L.D.: Multi instance neural networks. In: *Proceedings of ICML 2000, Workshop on Attribute-Value and Relational Learning* (2000)

9. Ray, S., Craven, M.: Supervised versus multiple instance learning: an empirical comparison. In: Proceedings of the 22nd International Conference on Machine Learning, ICML 2005, pp. 697–704. ACM, New York (2005)
10. Ray, S., Page, D.: Multiple instance regression. In: Proceedings of the Eighteenth International Conference on Machine Learning, ICML 2001, pp. 425–432. Morgan Kaufmann Publishers Inc., San Francisco (2001)
11. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: a lazy learning approach. In: Proceedings of the Seventeenth International Conference on Machine Learning, ICML 2000, pp. 1119–1126. Morgan Kaufmann Publishers Inc., San Francisco (2000)
12. Weidmann, N., Frank, E., Pfahringer, B.: A two-level learning method for generalized multi-instance problems. In: Lavrač, N., Gamberger, D., Blockeel, H., Todorovski, L. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 468–479. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39857-8_42
13. Zhang, Q., Goldman, S.A.: EM-DD: an improved multiple-instance learning technique. In: Advances in Neural Information Processing Systems, pp. 1073–1080. MIT Press (2001)



Integrating a Framework for Discovering Alternative App Stores in a Mobile App Monitoring Platform

Massimo Guarascio^{1(✉)}, Ettore Ritacco¹, Daniele Biondo², Rocco Mammoliti²,
and Alessandra Toma²

¹ Institute for High Performance Computing and Networking of the Italian National Research Council (ICAR - CNR), Arcavacata, Italy
massimo.guarascio@icar.cnr.it

² Poste Italiane, Rome, Italy

Abstract. Nowadays, implementing brand protection strategies has become a necessity for enterprises delivering services through dedicated apps. Increasingly, malicious developers spread unauthorized (fake, malicious, obsolete or deprecated) mobile apps through alternative distribution channels and marketplaces. In this work, we propose a framework for the early detection of these alternative markets advertised through social media such as Twitter or Facebook or hosted in the Dark Web. Specifically, it combines a data modeling approach and an ensemble learning technique, allowing to recommend web pages that are likely to represent alternative marketplaces. The framework has been implemented in a prototype system called Unauthorized App Store Discovery (UASD), and integrated in a security enterprise platform for the monitoring of malicious/unauthorized mobile apps. UASD allows to analyze web pages extracted from the Web and exploits a classification model to distinguish between real app stores and similar pages (i.e. blogs, forums, etc.) which can be erroneously returned by a common search engine. An experimental evaluation on a real dataset confirms the validity of the approach in terms of accuracy.

1 Introduction

Recently, an increasing attention has been paid to the problem of implementing adequate strategies for defending the company's brands from an unauthorized use. In particular, the brand reputation represents an open issue for all the enterprises delivering services through dedicated apps. Indeed, malicious developers frequently spread counterfeit apps which can compromise the security of the user devices and eventually the reputation of the original developer. Nevertheless, in this complex scenario, smartphones and tablets have become the devices mainly used by millions of users which daily install a wide range of applications provided through marketplaces and app stores. Therefore, it's not unusual that unaware users install unauthorized (e.g. fake, malicious, obsolete and deprecated) apps,

which can potentially harm the device and consequently the brand reputation of the copyright owner. Several technical reports, drafted by important security software companies, show an exponential growth of virus and trojans able to attack commonly used devices¹. It is quite easy to develop variants of well-known malware², and many popular security tools are not able to counteract common malware transformation techniques [16].

In many cases, these malicious programs are disguised as popular apps spreading via official or alternative market places (e.g. Amazon app store for Android³). As an example⁴, in July 2016 approximately two hundred mobile apps were diffused as an official version of the *Pokémon Go* game,⁵ most of which being malicious applications able to permanently lock the device. Other examples are the *PayPal App* clones and counterfeits which aim at stealing the login data of unaware users who accidentally update the original app to the malicious version.

Besides representing a risk for the end user, the spread of these applications represents a serious threat even for the original service providers and developers. Being directly or indirectly associated with harmful apps can cause users to give up on their services, thus causing a reputational damage. That's why the early detection of these apps is becoming strategic from a *Brand Protection* point of view [20]. Notice that the problem is relevant even in situations where there are no security issues, but just the possibility of being associated with poorly designed apps and services. For example, the mobile app *BancoSaldo* is a third-party app providing many utilities for handling financial services of the Italian postal service "Poste Italiane". The whole app (UI and Logo) has been designed to look a legitimate Poste Italiane product. Hence it's easy for an end user to confuse this app for an official one distributed by Poste Italiane, thus attributing all the potential failures to Poste Italiane itself.

The problem is that typically, malicious, fake or obsolete apps spread via unofficial channels (e.g. *alternative marketplaces and app stores*) accessible both via regular and *Dark Web* and therefore they are difficult to be discovered. The capability of identifying and monitoring alternative marketplaces both on Regular and *Dark Web* is a relevant and challenging task. These marketplaces are extremely dynamic and often do not monitor the published apps, and typically, they are advertised through social media posts. Notably, companies set up teams of experts and specialized personnel to discover these alternative markets and to inspect whether potentially harmful apps are available which can be associated with them. Anyway, the whole process to detect unauthorized app stores is usually performed manually by exploiting suitable queries on well-known research engines (e.g. by employing google hacking techniques) and they strongly rely on the skills of the operator.

¹ <https://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2012.pdf>.

² <http://www.intel.se/content/dam/www/public/us/en/documents/reports/mcafee-threats-quarterly-report.pdf>.

³ <https://www.amazon.com/mobile-apps/b?ie=UTF8&node=2350149011>.

⁴ <http://www.silicon.co.uk/security/fake-pokemon-go-mobile-apps-195141>.

⁵ <http://www.pokemongo.com/>.

Defining semi-automatic monitoring protocols is crucial for effectively counteracting the malicious mobile app diffusion, since it allows human operators to analyze a wider web search space. Our main aim, in this work, is to propose an *intelligent* infrastructure for continuously monitoring and analyzing the Web in order to detect alternative or unofficial marketplaces that may contain unauthorized mobile apps.

The current literature has mainly focused on the problem of monitoring and detecting malicious mobile apps, and little effort has been devoted to the detection of alternative markets.

In this work we define a semi-automatic approach for helping experts and specialized personnel to discover the alternative marketplaces available both on Regular Web and TOR (*The Onion Router*) Network. The learning method allows to assign a *score* to the web pages returned by performing the operator queries and then provides a sorted list according to the probability that a page is actually an appStore. The approach has been implemented in a prototype system called *Unauthorized App Store Discovery (UASD)*. Notably, UASD is integrated into the *Mobile Apps Security Monitoring (MASM*⁶) infrastructure, a security platform for monitoring malicious apps which is exploited by Poste Italiane.

To the best of our knowledge this is the first attempt to tackle this problem. To summarize, the main contributions of this work are:

- A prototype system for proactively discovering (alternative) app stores on Regular and Dark Web;
- A learning approach for accurately classifying and ranking web pages according to the probability to be a real mobile apps marketplace (*UASD*).

The rest of the paper is organized as follows. After introducing some preliminary concepts in Sect. 2, we present our solution approach for discovering alternative app stores. Section 3 introduces the logical architecture of MASM and how UASD has been integrated for improving the capability of the platform in detecting malicious mobile apps. An empirical analysis on a real case study is then discussed in Sect. 4. Finally, the Sect. 5 concludes the paper and presents some interesting future developments.

2 The UASD Framework

UASD stands for *Unauthorized App Store Discovery*, and it is a semi-automatic machine learning approach that supports human operators recommending the most likely alternative app stores in a (Dark) Web research space. *UASD* is composed by three main macro components shown in Fig. 1: Information Retrieval, Knowledge Discovery and Interaction with the operator. The details of the modules are as follows.

⁶ http://www.posteitaliane.it/en/innovation/technology_centre/certcyb.shtml.

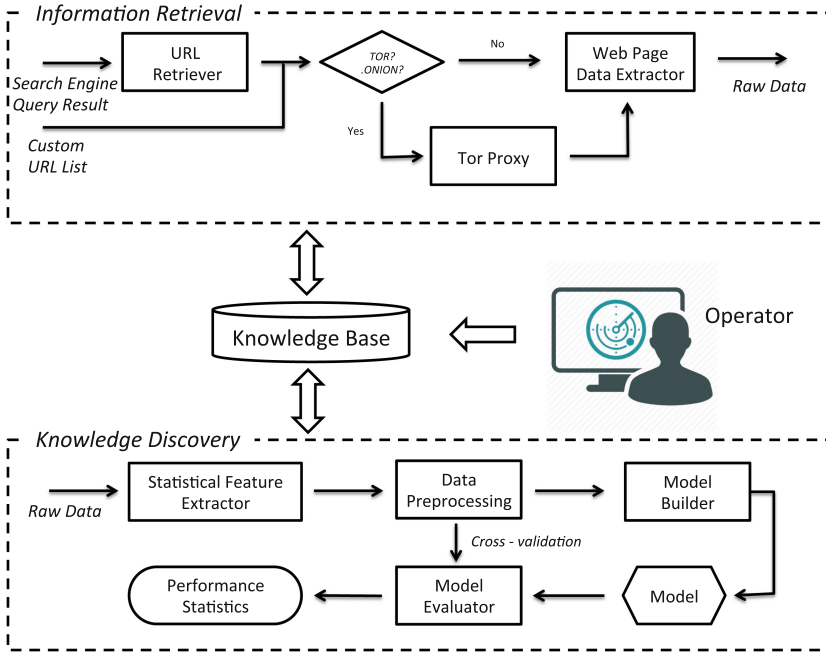


Fig. 1. Logical flow of UASD.

2.1 Information Retrieval

Human experts devise a set of web queries by exploiting Advanced Google Search Operators⁷ or specify some URLs in order to identify possible alternative mobile markets in regular Web or in the *TOR* network (Dark Web). Typically, these queries contain keywords referring to the names of the most popular mobile operative systems, app categories (such as e.g. “arcade games”, “puzzles”), specific mobile apps or even mobile devices. The **Information Retrieval** module is fed with these queries. Its task is to interpret them and extract the related pages. The **URL Retriever** retrieves the URLs from the query results. If a URL belongs to the *TOR* network, then a **Tor Proxy** allows the system to access the page in the dark web and collect its content. The platform allows to integrate different strategies for evaluating if an URL belongs to Tor Network, the most simple method is checking it ends with “*onion*”. At the end of this process a **Knowledge Base** is populated with all the extracted pages in HTML format. The **Web Page Data Extractor** allows to store in the Knowledge Base the html text (raw data) contained in the selected web pages.

⁷ https://en.wikipedia.org/wiki/Google_hacking.

2.2 Knowledge Discovery

This component allows to learn a classification/prediction model from the data (knowledge base) gathered by the information retrieval module. We can devise three components in it.

Data Transformation. Collected data need to be transformed and filtered in order to be provided as input to the machine learning process. A crucial step for this task is performed by a **Statistical Feature Extractor**, that allows to devise a set of (discriminative) structural features for each web page, denoted as **embedded attributes**. These features are based on the assumption that the Web can be modeled as a graph whose nodes are the web pages and edges are the hyperlink references. Each node has a neighborhood composed by those nodes that are directly linked to it. Hence, given a raw web page, the current implementation of the Statistical Feature Extractor builds a set of discriminative features (embedded attributes) based on target page’s neighborhood. Table 1 summarizes the main features extracted from the pages.

Table 1. Embedded attributes.

Attribute	Description
<code>isTorLink</code>	A boolean flag highlighting whether the page was extracted from the Dark Web
<code>NumberIntraDomainLinks</code>	The number of neighbors within the same web domain of the target page
<code>IntraDomainLinkPercentage</code>	The ratio between the <code>NumberIntraDomainLinks</code> and the neighborhood size
<code>IntraDomainDistinctLinkPercentage</code>	Similar to <code>IntraDomainLinkPercentage</code> but distinct links are considered
<code>NumberDownloads</code>	Number of direct download links for apps (e.g. <i>.apk</i>) or keywords (e.g. <i>install</i>) that allow a user to get a mobile application
<code>NumberKeywords</code>	Number of keywords typically included in app stores (suggested by domain experts): e.g. <i>android, apk, ios, access, app, categories, market, price, best, popular, top, rated, ios, windows phone</i> , etc.
<code>NumberDownloadFirstLevel</code>	The cumulative <code>NumberDownloads</code> for the neighbors within the same domain of the target page
<code>NumberKeywordsFirstLevel</code>	The cumulative <code>NumberKeywords</code> for the neighbors within the same domain of the target page

The embedded attributes allow to exploit the frequent (structural) patterns featuring the app store pages, e.g. typically an app store will contain several links for downloading *.apk* files in the home page or in the pages directly linked from the home page. It is worth noting that, in many cases keeping both `IntraDomainLinkPercentage` and `IntraDomainDistinctLinkPercentage` allows to exploit more information: for example, despite some malicious web sites could have the same `IntraDomainLinkPercentage` values as legal ones, the `IntraDomainDistinctLinkPercentage` value is much lower because they try to deceive the user by redirecting him to the (same) fraudulent service. If the `IntraDomainLinkPercentage` and the `IntraDomainDistinctLinkPercentage` exhibit similar values for the same page, the page is likely a real marketplace or a page with a similar structure (e.g. blog, forum, etc.).

Besides the embedded attributes, content features can be extracted directly from the HTML code of the page. Specifically, the page is described by the *META* tags and the keywords belonging to its *BODY* (where HTML tags are removed). The **Data Preprocessing Module** filters such a content and produces further attributes: specifically, *META* tags are directly converted in (logical) relational attributes (Bag Of Words model), text extracted from the *BODY* is converted by exploiting *Text Mining* techniques.

The text mining process is composed by the following steps:

- **Tokenization.** The text is separated by words and filtered from characters different from the english alphabet (included numbers, punctuation and special characters as “&”, “\”, ...).
- **Stop word removal.** The most common words in a language (conjunctions, adverbs, prepositions, ...) are removed, since, typically, their discriminative capability of distinguishing between two or more classes (i.e. categories) is very poor.
- **Stemming.** Words are reduced to their *word stem*, a concept similar to the base or root of a word. The stem doesn’t need to be identical to the morphological root of the word: it usually suffices that related words map to the same stem. For instance, the words “fishing”, “fished” and “fisher” share the same stem “fish”.
- ***N*-Gram generation.** An *N*-Gram is a contiguous sequence of *N* words in a text. The *N*-Gram generation process builds all possible *N*-Grams from the filtered *BODY* of the pages. For instance, consider the sentence “The quick brown fox jumps”, it has three 3-grams: “The quick brown”, “quick brown fox” and “brown fox jumps”. The Data Processing module extracts *N*-Grams with $N \in \{1, 2, 3\}$.

The merging of embedded attributes, meta tags and textual features represents the general schema upon which to characterize an extracted web page. Notice that the resulting table exhibits very high dimensionality and a huge sparseness factor. The Data Preprocessing Module is also demanded to perform a feature selection routine based on χ^2 test, *tf-idf*, correlation [5], and *PCA* [6]. According to these measures, the attributes with low discriminative power are discarded and the dimensionality of the dataset is strongly reduced. Finally,

a small portion of the cleaned data is manually labeled by experts as either **app store** or **Regular Web**.

Prediction Model. The output of the Data Preprocessing Module is a *Cleaned Dataset* that can be used for generating a classifier capable to discriminate between records corresponding to actual app store pages and records referring to regular Web. UASD relies on an Ensemble classifier [7].

An Ensemble is a combination of two or more classifiers (typically at least three) according to different strategies in order to achieve a better prediction: the idea is that classification errors are less likely with several classifiers rather than a single classifier. In literature different combination strategies have been proposed, the most known of them are three: Bootstrap aggregation (bagging) [1], Boosting [18] and Stacking [21].

- *Bootstrap Aggregating.* The Bagging strategy randomly draws different subsets from the data set; each subset is exploited to build a classifier. The classifier set is demanded to vote about the prediction: if the majority of the classifiers votes for an outcome (e.g. “the page is an AppStore”) then the ensemble will predict that result.
- *Boosting.* Boosting defines an incrementally strategy to build an ensemble. Data feed a classifier that returns its prediction with a confidence level. Each record in the data is then equipped with a weight computed according to the prediction: the bigger the error in terms of misclassification and confidence, the bigger the weight. A new classifier is built exploiting the weighted data: it will focus on those tuples with higher weight trying to correctly classify them. The new classifier can overwrite the weights to generate another classifier. This loop can continue until some stopping criterion is reached. The final classifier is the result of the ensemble.
- *Stacking.* The Stacking approach, also called stacked generalization, is a two-step strategy. In the first step, several different classifiers, *Base Learners*, are trained on the whole data set: each one of the will enrich the data with its prediction. A new data set, often called *Stacked-View*, is then built by the combination of all the predictions. The stacked-view is provided as input for one last classifier, called meta-classifier. The prediction process of a new record will follow these two steps: firstly, the record will be equipped with the base learners’ predictions, then the meta-classifier returns the final prediction about the enriched record.

In our solution, we use a stacking approach since the amount of labeled data is limited (as detailed in Sect. 4). Small data sets are more prone to the overfitting problem and it is well known from the literature that stacking is more robust in this setting.

The structure of the stacking method adopted in UASD is shown in Fig. 2. We chose 9 different base learners, in order to promote the diversity of the single predictions. These base learners are algorithms which work particularly well in unbalanced scenarios (i.e. where a dominant class exists). This is exactly the situation we cope with: the frequency of apps stores in the training data

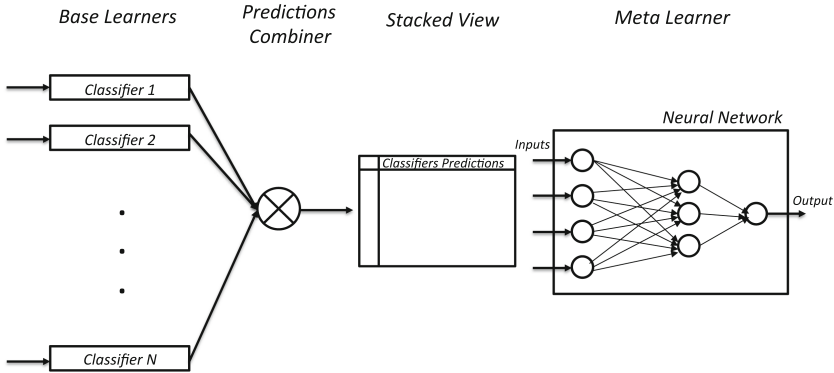


Fig. 2. UASD ensemble model.

is extremely lower than the regular web pages' one. The complete list of the chosen algorithms is the following: *AODE* [19], *Hidden Naive Bayes* [22], *Maximum Entropy* [13], *Mine Rule AODE* [3], *Mine Rule Naive Bayes* [3], *Bayesian Approach (Discretization)* [9], *Bayesian Approach (Kernel Transformation)* [9], *Nearest Neighbors* [4] and *Logistic Regression* [12].

The predictions of these base learners are combined in a stacked view (where each column contains the prediction provided by a single classifier) that feeds an Artificial Neural Network (ANN) as a meta-learner. In this way, the output of the ANN will be a new probability value obtained as a suitable combination of the probabilities provided by the base models. Hence, it will be exploited as score to rank the sites from the most likely app store to least. Using ANNs for combining the predictions provided by different classifiers has already successfully exploited [8] in many challenging scenarios. For example, the Otto Group Product Classification Challenge⁸ has been won by a stacking model composed of over 30 base learners whose output was provided as input for three meta-classifiers: XGBoost, Neural Network, and Adaboost. In particular, we chose an ANN because it can effectively and efficiently handle the great number of attributes of the Cleaned Dataset enriched by the base learners' predictions.

Evaluation. We assess the quality of the devised predictive engine by computing some accuracy metrics. As aforesaid, the discovering of novel app stores is an unbalanced classification problem: the number of positive examples (actual app store) is overwhelmed by the negatives (rest of the web pages). Different evaluation metrics have been proposed in literature for testing the effectiveness of classification models in presence of a rare class. Indeed, the usage of metrics that do not adequately accounts for the rarity of such a minority class may lead to overestimating the real capability of a classifier to correctly recognize the instances of that class. Typically, some core metrics are based on the following

⁸ <https://www.kaggle.com/c/otto-group-product-classification-challenge/discussion/14335>.

statistics: (i) *True Positives (TP)*, i.e. the number of positive cases correctly classified as such; (ii) *False Positives (FP)*, i.e. the number of negative cases incorrectly classified as positive; (iii) *False Negatives*, i.e. the number of positive cases incorrectly classified as negative; and (iv) *True Negatives*, i.e. the number of negative cases correctly classified as such.

Classification *accuracy* is the fraction of cases classified correctly: $(TP + TN)/(TP + FP + FN + TN)$. Despite this is a widespread evaluation metric, it is not appropriate when the classes are imbalanced. For instance, in a scenario where only 1% of tuples belongs to the minority class, a simple model that predicts every instance as belonging to the majority class would have an accuracy of 99%, although it is not able to recognize the class of interest (in our case, the app stores).

We exploit such statistics to compute the standard *Precision (P)* and *Recall (R)* measures [2] on the minority class, in order to support fine grain analyses on the misclassification errors made over those instances:

$$Prec = \frac{TP}{TP+FP}$$

$$Rec = \frac{TP}{TP+FN}$$

The F-measure, defined as

$$F_1 = \frac{2 * Prec * Rec}{Prec + Rec}$$

represents the harmonic mean of the above measures and it is used to combine them in a single score [17].

2.3 Human Interaction and Incremental Learning

The set up phase is the initial process of the system, during which the first model is generated. This phase requires a small seed set of manually labeled web pages by a team of experts. After this preliminary step, the system works in a semi-supervised way. The overall process is monitored by human experts, as described in Fig. 3 which represents the steady and operational states of the system. The operator defines a set of queries and selects a set of target links to submit to the URL Retriever. Relevant data is extracted and preprocessed from the links, and the resulting pages are submitted to the prediction engine which performs the prediction and ranks the pages according to the score provided by the prediction model (see Sect. 2.2). The list is returned to the operator, then her feedbacks are again exploited to enrich the original training data and consequently update the prediction model. Obviously, the whole procedure can be computationally and temporally expensive (according the number of queries submitted to the system) due the human interaction, therefore a pipe-line scheme is used: the result of a pre-processed set of queries is provided to the operators while another set is in the processing phase. In this way, the extracting, storing and analyzing steps are always performed off-line without introducing latency.

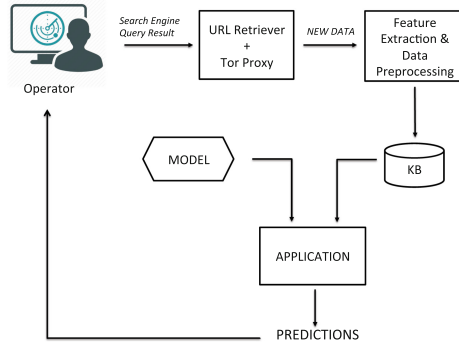


Fig. 3. Interaction with the operator.

3 Mobile App Security Monitoring

In this section we describe MASM, a security enterprise platform for discovering and monitoring (malicious) apps available in official and alternative market-places. MASM integrates UASD and exploits the app stores discovered by the latter to detect potentially harmful apps which counterfeit the company brand.

MASM is an advanced platform designed to support the operator in the whole process for detecting malicious behaviors of mobile apps and can be effectively employed to monitor the unauthorized usage of the brand of a company. The (security) monitoring methodology implemented in MASM consists of 5 main phases: *Mobile App Detection, Analysis, Mitigation and Reporting, Shutdown, Regular Checking*. Figure 4 shows the logical flow of the approach.

App Detection is the first step of this process and it provides a set of methods and tools to identify the unauthorized usage of the Brand to protect. In particular, this layer searches on the Web applications referring to the Brand in their name or description. It allows to discover the (unauthorized) apps by exploiting techniques of image analysis and comparison on their logos.

An application deemed as relevant undergoes an analysis process aiming at identifying several types of security issues. The *Analysis* step is based on the three-tier approach summarized below:

- Static analysis is performed without actually executing the app. Specifically, it is performed by analyzing the application “as-is” (mainly using code inspection techniques)
- Dynamic analysis aims at studying the run-time behavior of the apps and it is performed executing them in a controlled environment (a *sandbox*). This type of analysis focuses on the interactions between the smartphone and other entities (e.g. web servers, local/remote files, other apps, etc.).
- Finally, *legal* analysis is performed which aims at identifying violations of contractual obligations or laws. The results of this analysis are shared with corporate functions within the Directorate of Legal Affairs.

The *Mitigation and Reporting* phase provides a detailed report of the analysis conducted and defines suitable strategies for the mitigating and/or contrasting the identified security issues. *Shutdown* phase takes severe actions against malicious/unauthorized mobile apps (like, e.g., forward remove requests to the marketplaces for the deletion of these apps). Finally, the *Regular Check* phase allows continuous monitoring of apps detected and analyzed.

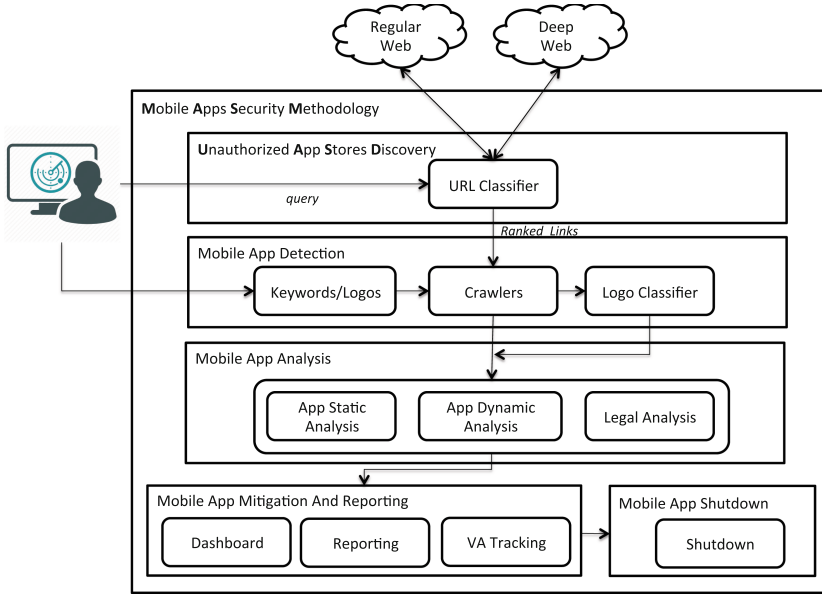


Fig. 4. Mobile Apps Security Methodology (MASM) - logic architecture

With regard to the logic architecture shown in Fig. 4, UASD represent the topmost layer, and it supports the initial phase of discovering unknown app stores.

4 Experimental Evaluation

In this section, we discuss the experiments carried out on a real application scenario. The scenario and the data at hand are discussed first. We next discuss the setting adopted to evaluate the quality of discovered models and evaluate the results of the performed test.

Case Study. The evaluation is performed on a crawl of 440 pages examined by domain experts at Poste Italiane. Within these, 40 web pages represent actual app stores and the remaining pages represent normal sites. The pages underwent the preprocessing module and a final dataset characterized by the feature

described in Sect. 2 was created. The dataset clearly devises an unbalanced classification problem where the number of examples, belonging to the class of interest, is significantly lower than the majority class. The data preparation phase generates almost 12,000 attributes (mostly textual features and META tags), which are further preprocessed by the feature evaluator and consequently reduced to approximately 3,000 relevant attributes.

Performance Measures and Evaluation Setting. The model evaluation is accomplished relying on Stratified 10-fold Cross Validation [15], and measuring Precision (i.e., confidence of a prediction), Recall (i.e., coverage of a prediction) and F-Measure (the harmonic mean between precision and recall, also called *F1-Score*) [14].

Experimental Results. The purpose of the analysis is twofold.

- First, we aim at evaluating the role of the embedding attributes. Under this perspective, we evaluate the performance of the base classifiers by considering the full set of features and by excluding the embedding attributes.
- Second, we evaluate the role of the stacking architecture. In principle, the addition of a further level where the set of available features is enriched should allow to better balance the prediction.

In the context we are analyzing, the objective is to obtain a good balance between precision and recall with regard to the minority class label. In fact, the aim is to increase the number of covered marketplaces by simultaneously minimizing the number of false positives, i.e. the incorrect suggestions to the

Table 2. Evaluation of $F_1 - Score$, precision and recall on base models compared to UASD.

Use embedded attributes	Model	F ₁ -Score	Precision	Recall
N	AODE	0,712	0,788	0,650
	Hidden Naive Bayes	0,687	0,852	0,575
	Max-Ent	0,693	0,743	0,650
	Mine Rule AODE	0,699	0,674	0,725
	Mine Rule Naive Bayes	0,658	0,694	0,625
	Bayesian Approach (Discretization)	0,692	0,711	0,675
	Bayesian Approach (Kernel Transformation)	0,687	0,917	0,550
	Nearest Neighbors	0,667	0,846	0,55
	Logistic Regression	0,694	0,781	0,625
	Y	AODE	0,740	0,818
Hidden Naive Bayes		0,722	0,813	0,650
Max-Ent		0,707	0,690	0,725
Mine Rule AODE		0,690	0,659	0,725
Mine Rule Naive Bayes		0,700	0,700	0,700
Bayesian Approach (Discretization)		0,725	0,725	0,725
Bayesian Approach (Kernel Transformation)		0,725	0,862	0,625
Nearest Neighbors		0,694	0,781	0,625
Logistic Regression		0,743	0,867	0,650
<i>UASD Approach</i>	0,757	0,824	0,700	

operator. In this respect, $F_1 - Score$ represents a good choice to evaluate the performance of the classification model.

In Table 2 we report the results of this analysis. The role of the embedding attributes can be clearly appreciated and it increases the general performance of the base classifiers by 5%. The stacker is also a winner in the evaluation, as its adoption further boost the prediction quality.

4.1 UASD in Action

UASD is currently employed by a Poste Italiane expert team to monitor the introduction of potentially harmful apps. The platform, integrated in MASM⁹ an advanced security system for the analysis of mobile apps, provides the operator with a set of graphical tools for querying the recommendation engine which returns a ranked list of URLs ordered according the estimated probability to be a market.

In Fig. 5, we show an example screenshot, where the result of a specific query returns a set of potentially matching (unknown) marketplaces. Within the figure,

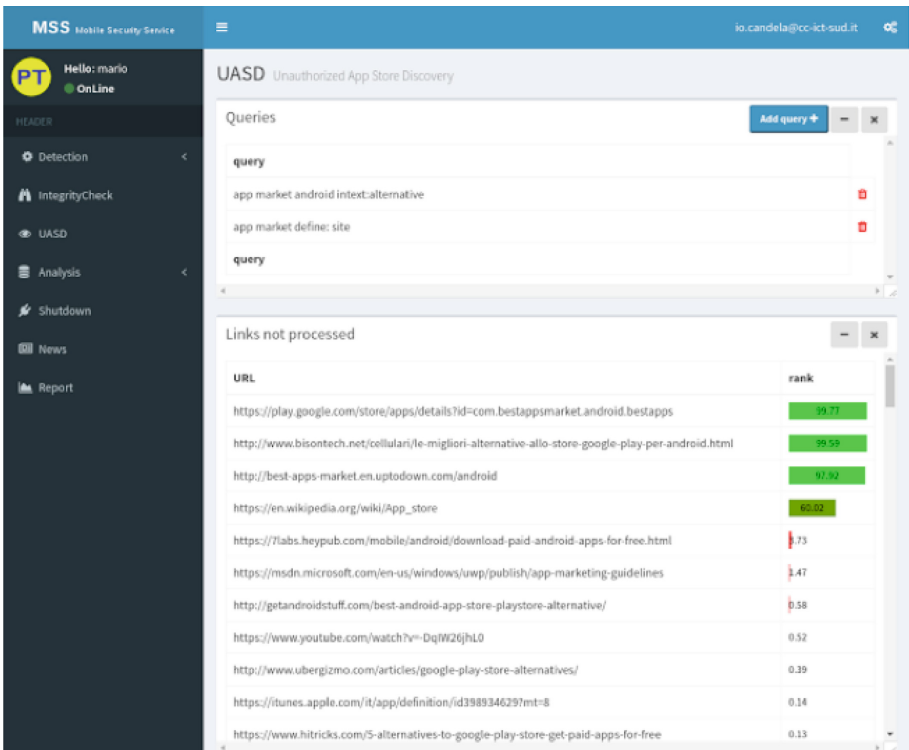


Fig. 5. Dashboard of MASM integrating UASD.

⁹ http://www.posteitaliane.it/en/innovation/technology_centre/certyb.shtml.

we can notice the first four matches, representing three true positive examples (with a probability above 97%) and a false positive example (the wikipedia page for “app store” deemed as an app store with probability 60%). It is interesting to highlight that the latter contains many terms that typically characterize an actual marketplace and exhibits a similar structure in terms of intra-domain links, and at the same time the lack of other embedded features (such as direct download links to apk) lowers the positiveness of the likelihood.

5 Conclusion

Detecting malicious behaviors of mobile apps is a challenging task and it comprises a continuous monitoring of the apps available in marketplaces. Most of the current approaches assume prior knowledge about the marketplaces to be monitored. Due to the continuous growth and dynamism of mobile app providers, identifying these app stores is becoming a difficult and time-consuming task. In this work, we propose a semi-automatic machine learning approach based on a suitable set of (derived) discriminative features and an ensemble learning method for discovering alternative mobile app marketplaces. Our approach has been implemented in a prototype platform for the proactive searching of marketplaces both on Regular and Dark Web, called UASD.

UASD provides a list of URLs (extracted from a set of queries defined by the Human operator) sorted according the probability to be an actual app store, then, the operator can evaluate the provided URL list. The platform has been integrated as a service of MASM, an advanced security system for the analysis of mobile apps that is developed and currently employed by Poste Italiane. Experimental findings on a real use case confirm that our approach is effective in identifying these marketplaces.

As future work, we plan to investigate the possibility to use Deep Learning approaches [10] to automatically discover higher-level features from raw data. This could allow to extract further discriminative features that are difficult to be manually defined. Moreover, we want to investigate the possibility to equip our approach with some collective mining techniques, e.g. [11], in order to exploit other informations related to the structure of the web page links.

References

1. Breiman, L.: Bagging predictors. *Mach. Learn.* **24**(2), 123–140 (1996)
2. Buckland, M., Gey, F.: The relationship between recall and precision. *J. Am. Soc. Inf. Sci.* **45**(1), 12–19 (1994)
3. Costa, G., Guarascio, M., Manco, G., Ortale, R., Ritacco, E.: Rule learning with probabilistic smoothing. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) *DaWaK 2009*. LNCS, vol. 5691, pp. 428–440. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03730-6_34
4. Cover, T., Hart, P.: Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.* **13**(1), 21–27 (2006)

5. Hall, M.A.: Correlation-based feature selection for machine learning. Technical report (1999)
6. Jolliffe, I.T.: *Principal Component Analysis*, 2nd edn. Springer, New York (2002). <https://doi.org/10.1007/b98835>. <http://www.worldcat.org/isbn/0387954422>
7. Jurek, A., Bi, Y., Wu, S., Nugent, C.: A survey of commonly used ensemble-based classification techniques. *Knowl. Eng. Rev.* **29**(5), 551–581 (2014)
8. Koehn, P.: Combining multiclass maximum entropy text classifiers with neural network voting. In: Ranchhod, E., Mamede, N.J. (eds.) *PorTAL 2002*. LNCS (LNAI), vol. 2389, pp. 125–131. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45433-0_19
9. Langley, P., Iba, W., Thompson, K.: An analysis of Bayesian classifiers. In: *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI 1992, pp. 223–228. AAAI Press (1992)
10. Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
11. Loglisci, C., Appice, A., Malerba, D.: Collective regression for handling autocorrelation of network data in a transductive setting. *J. Intell. Inf. Syst.* **46**(3), 447–472 (2016)
12. McCullagh, P., Nelder, J.A.: *Generalized Linear Models*, 2nd edn. Chapman & Hall, London (1989)
13. Phillips, S.J., Dudík, M., Schapire, R.E.: A maximum entropy approach to species distribution modeling. In: *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML 2004, p. 83. ACM, New York (2004)
14. Powers, D.M.W.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *J. Mach. Learn. Technol.* **2**(1), 37–63 (2011)
15. Purushotham, S., Tripathy, B.K.: Evaluation of classifier models using stratified tenfold cross validation techniques. In: Krishna, P.V., Babu, M.R., Ariwa, E. (eds.) *ObCom 2011*. CCIS, vol. 270, pp. 680–690. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29216-3_74
16. Rastogi, V., Chen, Y., Jiang, X.: Catch me if you can: evaluating android anti-malware against transformation attacks. *Trans. Inf. Forensics Secur.* **9**(1), 99–108 (2014)
17. van Rijsbergen, C.J.: *Information Retrieval*, 2nd edn. Butterworth-Heinemann, Newton (1979)
18. Schapire, R.E.: The strength of weak learnability. *Mach. Learn.* **5**(2), 197–227 (1990)
19. Webb, G.I., Boughton, J.R., Wang, Z.: Not so Naive Bayes: aggregating one-dependence estimators. *Mach. Learn.* **58**(1), 5–24 (2005)
20. Wilson, J.M.: *Brand protection 2020*. Technical reports, Michigan State University (2015)
21. Wolpert, D.H.: Stacked generalization. *Neural Netw.* **5**, 241–259 (1992)
22. Zhang, H., Jiang, L., Su, J.: Hidden Naive Bayes. In: *Proceedings of the 20th National Conference on Artificial Intelligence*, AAAI 2005, vol. 2. AAAI Press (2005)



Usefulness of Unsupervised Ensemble Learning Methods for Time Series Forecasting of Aggregated or Clustered Load

Peter Laurinec^(✉)  and Mária Lucká

Faculty of Informatics and Information Technologies,
Slovak University of Technology in Bratislava,
Ilkovičova 2, 842 16 Bratislava, Slovak Republic
{peter.laurinec,maria.lucka}@stuba.sk

Abstract. This paper presents a comparison of the impact of various unsupervised ensemble learning methods on electricity load forecasting. The electricity load from consumers is simply aggregated or optimally clustered to more predictable groups by cluster analysis. The clustering approach consists of efficient preprocessing of data gained from smart meters by a model-based representation and the K-means method. We have implemented two types of ensemble learning methods to investigate the performance of forecasting on clustered or simply aggregated load: bootstrap aggregating based and the newly proposed clustering based. Two new bootstrap aggregating methods for time series analysis methods were newly proposed in order to handle the noisy behaviour of time series. The smart meter datasets used in our experiments come from Ireland and Slovakia, where data from more than 3600 consumers were available in both cases. The achieved results suggest that for extremely fluctuate and noisy time series unsupervised ensemble learning is not useful. We have proved that in most of the cases when the time series are regular, unsupervised ensemble learning for forecasting aggregated and clustered electricity load significantly improves accuracy.

Keywords: Load forecasting · Clustering · Bagging
Ensemble learning

1 Introduction

Modern information technologies produce a large amount of data that can be used for further analysis, giving important insights into data that supports making informed decisions. One important source of data is smart meters - sensors measuring electricity consumption or production. A smart grid is an ecosystem created from smart meters that can control or monitor electricity load (of consumers) or both load and production (of prosumers), collecting a large amount of data and making interventions when needed. To make these interventions

or decisions useful, they must be supported by information provided by smart meter data. One of the key tasks in a smart grid is electricity load forecasting, which is essential for energy distribution and utility companies, salesmen and for end users. Developing more sophisticated and accurate forecasting methods is important and for these purposes, data mining and machine learning methods are developed and adapted.

There are several suitable methods for load forecasting such as time series analysis and regression methods. Both types of them have their limitations such as an inability of adaptation to sudden changes (concept drift) and the noisy behaviour of time series. Therefore to find and choose the most suitable forecasting method is difficult. Besides already existing forecasting methods, there is a promising approach using the proper combination of various methods overcoming limitations of particular ones. This method is called ensemble learning. Moreover, forecasting methods themselves can be tuned by the simplest of all ensemble methods - bootstrap aggregating (bagging).

We have evaluated two types of methods of bagging that are based on type of the used forecasting method: (a) time series analysis method; (b) regression tree. Three types of bootstrapping methods were implemented to observe their usefulness for time series analysis methods: (a) moving block bootstrap with combination of STL decomposition and Box-Cox transformation; (b) smoothed version of the previous one; (c) K-means based bootstrapping. Bagging for regression trees was the classical sampling with replacement combined with randomized values of hyperparameters. We have proposed several new ideas to unsupervised ensemble learning approaches relying on a proper combination of multiple bootstrap forecasts. Their advantages and disadvantages were discussed in our work.

Another very important but totally different approach to optimising forecast accuracy is based on advanced time series data mining methods. This includes cluster analysis that is used for consumer segmentation according to their consumption patterns, so more predictable groups of consumers are created. As we showed in our previous works [1,2], this approach has the promising improvement of forecasting accuracy.

The aim of this paper is to compare the combination of the time series data mining approach with the newly proposed ensemble learning methods for improving forecasting accuracy.

This paper is structured as follows: Sect. 2 contains an introduction with related works, while in Sect. 3 the datasets used in our experiments are described. Section 4 presents a description of our approach together with the methods used for time series processing, cluster analysis, forecasting and ensemble learning. Section 5 presents the description and the evaluation of performed experiments and the paper concludes with Sect. 6.

2 Related Work

Electricity load forecasting is a highly discussed research area due to the interesting character of data coming from smart meters. The time series of electricity

consumption can have various patterns and are affected by multiple seasonalities (daily, weekly and yearly), weather, holidays and other unexpected changes. For this reason, sophisticated machine learning methods are applied to tackle challenges linked with smart meter data.

Ensemble learning in load forecasting is a highly used method in solving the above-mentioned problems. Adhikari et al. [3] proposed a ranking based ensemble approach to incorporate only the best models to the final ensemble forecast. Shen et al. [4] proposed a pattern forecasting ensemble model, which combines forecasts created by clustering algorithms. Grmanová et al. [5] used median-based approach to optimise weights in the incremental heterogeneous ensemble learning model for consumption forecasting.

The usage of cluster analysis for more accurate forecasts of aggregated load is noted in the work of Shahzadeh et al. [6]. This paper deals with the clustering of consumers in three different ways of feature extraction from a time series. As a clustering method, K-means was used and the neural network has been applied as a forecast method. Wijaya et al. [7] used correlation-based feature selection as a representation of consumers in clustering, and linear regression, multi-layer perceptron and support vector regression were used as forecasting methods. In our previous work [2], four different representations of time series and ten forecasting methods were evaluated, in order to verify their suitability for the forecasting of clustered load. We have proved that optimised clustering of consumers significantly improves the accuracy of forecasts in combination with triple exponential smoothing, ARIMA, Random Forests and bagging.

Until now, the combination of clustering of consumers and ensemble learning has not been explored and evaluated. Therefore in the proposed paper we will (a) evaluate from two different types of bagging on basic forecasting methods and examine their behaviour on clustered load, (b) propose two new bootstrapping methods for time series to overcome noisy character of data, (c) design several new unsupervised clustering ensemble learning approaches for forecasting, (d) for clustering electricity consumers, propose efficient preprocessing through model-based representation of time series based on K-means clustering.

3 Smart Meter Data

For verification of our approach, we have used in our experiments two different datasets, comprising of data from smart meters. This data includes Irish and Slovak electricity consumption. The Irish data was collected by the Irish Commission for Energy Regulation (CER) and available from ISSDA¹ (Irish Social Science Data Archive). This data contains three different types of customers: residential, SMEs and others. The largest group is residential, where after removing consumers with missing data, we have 3639 residential consumers left. The frequency of data measurements was on a half-hour basis, during a day 48 measurements were performed. Slovak data was collected within the project “International Centre of Excellence for Research of Intelligent and Secure Information-

¹ <http://www.ucd.ie/issda/data/commissionforenergyregulationcer/>.

Communication Technologies and Systems". These measurements were obtained from Slovak enterprises, having a completely different nature than the Irish data. After removing consumers with missing data, those with zero consumption and consumption higher than 42 kW, the dataset comprised 3630 consumers. The frequency of data measurements was on a quarter-hour basis, so daily 96 measurements were performed. The frequency of data measurements was transformed to half-hourly in order to make it comparable with the Irish data.

The difference between the residential and enterprise data is significant. The amount of consumption in residences was low and not regular, as opposed to the enterprise, where the amount of consumption was very high and mostly regular during the week and irregular during the year (i.e., holidays for whole factory or in school, the period of year when central heating is turned on). Therefore different evaluation results for these two datasets could be expected.

4 Proposed Approach

Two approaches for the aggregation of electricity load were compared: based on clustering of consumers and based on simple aggregation. Moreover, three types of forecasting methods were compared: six basic methods, six basic methods with bagging and six ensemble methods. The clustering approach consists of these phases:

1. Normalisation of time series by z-score and calculation of a model-based representation of time series (estimation of regression coefficients). Extracted representations then enter a clustering method.
2. Calculation of an optimal number of clusters for given representations of a time series by DB-index. The actual clustering of consumers is followed by the K-means method.
3. Aggregation of consumption within the clusters and the application of a forecast model on training data. The forecast for the next period is calculated and aggregated. Finally it is compared with the real consumption.

This process is repeated incrementally until new data is available. The sliding window has length of 21 days and it is shifted by one day. It means that the oldest day from the window is removed and the data from a new day are added.

4.1 Clustering of Electricity Consumers

The first necessary step is the normalisation of the times series of electricity consumption by the z-score because we want to cluster similar patterns and not the time series according to the amount of energy consumption.

The next is the computation of the time series representation, which is an input to the clustering algorithm. The modification of the time series to its representation is performed by a suitable transformation. The main reason for using representations of time series is the pursuit of more effective and easier

work with time series, depending on the application. Using time series representations is appropriate because by reducing the dimension, it will reduce memory requirements and computational complexity, and it implicitly removes noise and emphasizes the essential characteristics of data. We conducted from our previous work that model-based representations are highly appropriate for seasonal time series [1]. For a model, multiple linear regression is used for extraction of regression coefficients of two seasonalities (daily and weekly). Formally, the model can be written as follows:

$$x_t = \beta_{d1}u_{td1} + \beta_{d2}u_{td2} + \dots + \beta_{ds}u_{tds} + \beta_{w1}u_{tw1} + \dots + \beta_{w6}u_{tw6} + \varepsilon_t,$$

for $t = 1, \dots, n$, where x_t is the t -th electricity consumption, $\beta_{d1}, \dots, \beta_{ds}$ are regression coefficients for daily season, s is the length of period of one day, $\beta_{w1}, \dots, \beta_{w6}$ are regression coefficients for a weekly season. Weekly regression coefficients are just six, not seven, because of prevention from singularity of the model. The $u_{td1}, \dots, u_{tdseas}$, u_{tw1}, \dots, u_{tw6} are independent binary (dummy) variables representing the sequence numbers in the regression model. They are equal to 1 in the case when they point to the j -th value of the season, $j = 1, 2, \dots, s$, in case of a daily season and $j = 1, 2, \dots, 6$ in case of a weekly season. The ε_t are random errors having the normal distribution of $N(0, \sigma^2)$ that are for different t mutually independent. The most widespread method for obtaining an estimate of the vector $\beta = (\beta_{d1}, \dots, \beta_{ds}, \beta_{w1}, \dots, \beta_{w6})$ is the Ordinary Least Squares method.

For grouping consumers into clusters, the centroid-based clustering method K-means with centroids initialization K-means++ [8] was used. The advantage over conventional K-means is based on carefully seeding of initial centroids, which improves the speed and accuracy of clustering.

In each iteration of a batch processing, we have automatically determined the optimal number of clusters to K using the internal validation rate Davies-Bouldin index [9]. The optimal number of clusters ranged from 8 to 18.

In Fig. 1 clustered time series representations of consumers from Slovakia are shown. We can see that the clusters 1, 2 and 8 have a similar daily pattern, but the weekly pattern is remarkably different, so our clustering approach is working correctly. As is apparent, other clusters are visibly different from each other.

4.2 Forecasting Methods

Basic Forecasting Methods. We have compared six basic forecasting methods to investigate their relevance in combination with bootstrapping, and to see if it benefits from clustering.

Seasonal decomposition of time series by Loess (STL) is a method that decomposes a seasonal time series into three parts: trend, seasonal and remaining [10]. For the resulting three time series, the result is used separately for the forecast with ARIMA model (STL+ARIMA) and Holt-Winters exponential smoothing (STL+EXP). The ARIMA model has been introduced by Box and Jenkins [11] and is one of the most popular approaches in forecasting.

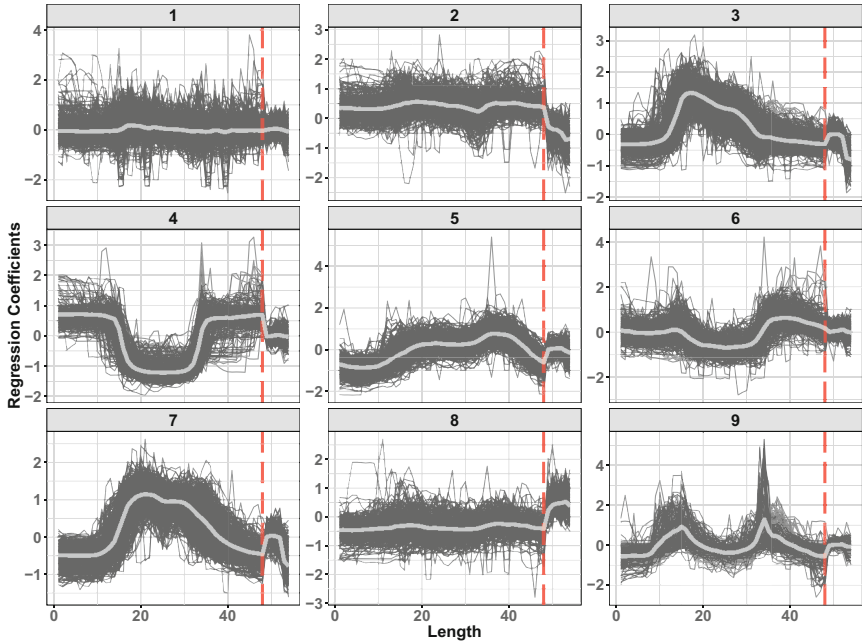


Fig. 1. Nine clusters of Slovak consumers. Grey line represents the centroid of a cluster.

The Holt-Winters exponential smoothing [12] is a forecasting method applied to a time series, whereby past observations are not weighted equally, but the weights decrease exponentially with time. The exponential smoothing method was also used stand-alone (EXP), but with one important enhancement for trend changing time series. Three different models were fitted each time and the best of them was picked to produce a forecast. These models were a full additive model with a trend component, an additive model with a damping trend and an additive model without a trend. The best model among them was chosen according to the best values of Akaike Information Criterion (AIC) [13].

Recursive partitioning regression trees that belong to Classification and Regression Trees methods (CART) search over all possible splits by maximising an information measure of node impurity, selecting the covariate showing the best split [14]. The most important hyperparameters that must be tuned are the minimum number of observations that are needed in node to split (set to 2), maximal depth of a tree (set to 30) and the complexity parameter (cp). The last parameter cp is a threshold deciding if each branch fulfils conditions for further processing (only nodes with fitness larger than factor $cp = 1E-6$ are processed).

For adaptation to a trend change, the dependent variable (time series of electricity consumption) was detrended by STL decomposition [10] in order to improve forecasting accuracy. From the extracted trend part of the time series, future values were forecasted by automatic ARIMA procedure [15] and added to the forecast from the CART model that predicts the aggregated seasonal and

remainder part of the time series. The attributes to the CART model are daily and weekly seasonal vectors. We considered the size of a daily period $s = 48$ and weekly period $w = 7$. Daily seasonal vector has the form $\mathbf{day}_j = (1, 2, \dots, s)$, $j = 1, 2, \dots, d$, where d is the number of days in the training window and $\mathbf{day} = (\mathbf{day}_1, \dots, \mathbf{day}_d)$. Let $\mathbf{i} = (i, \dots, i)$ is a vector of dimension s . Then the weekly seasonal vector has the form $\mathbf{week} = (\mathbf{1}, \mathbf{2}, \dots, \mathbf{w}, \mathbf{1}, \dots)$ and has the dimension of $s * d$.

Conditional inference trees (CTREE) is a statistical approach to recursive partitioning, which takes into account the distributional properties of the measurements [16]. Here the important hyperparameter for tuning is the minimal criterion that must be exceeded in order to implement a split (set to 0.925).

Two variants of CTREE method based on different attributes entering to the model were evaluated. The first one (CTREE.lag) has four seasonal attributes for daily and weekly periods in the sinus and cosinus form: $(\sin(2\pi \frac{day}{s}) + 1)/2$ resp. $(\cos(2\pi \frac{day}{s}) + 1)/2$, and $(\sin(2\pi \frac{week}{7}) + 1)/2$ resp. $(\cos(2\pi \frac{week}{7}) + 1)/2$. Another attribute for the model is the seasonal component of STL decomposition with a one day lag. The second one (CTREE.dft) uses as attributes two seasonal Fourier terms. As we found experimentally the best results were achieved with four terms for daily period $(\sin(\frac{2\pi jt}{48}), \cos(\frac{2\pi jt}{48}))_{j=1}^4$, and eight pairs of terms for weekly seasonality $(\sin(\frac{2\pi jt}{7}), \cos(\frac{2\pi jt}{7}))_{j=1}^8$, where $t = (1, \dots, n)$. Before using both CTREE variations, the original time series was also detrended by STL decomposition as was with CART method.

Bootstrap Aggregating Methods. Bootstrap aggregating (bagging) is an ensemble meta-algorithm [17], which creates multiple versions of a learning set to produce a multiple number of predictors. These predictors are then aggregated, for example by arithmetic mean. We have implemented two different types of bagging methods in order to adapt to two different types of forecasting methods: regression trees and time series analysis methods.

Classical bagging proposed by Breiman [17], generates multiple training sets by uniformly sampling the original one with replacement with some sample ratio. In our approach, the sample ratio and hyperparameters mentioned in the previous section concerning regression trees were also randomised. The sample ratio was randomly sampled in the range of 0.7 – 0.9. The CART hyperparameters were sampled this way: maximal depth in range of 26 – 30, minimal split 2 – 3 and cp 9E-7 – 1E-5. The CTREE hyperparameter minimal criterion is sampled 0.88 – 0.97. Each time 150 trees were created and the resulting forecasts were aggregated by median.

For the time series analysis methods (STL+ARIMA, STL+EXP, EXP), the bagging proposed by Bergmeir et al. [18] was used. At first a Box-Cox transformation to the data was applied, then the series was decomposed into three components by STL. The remainder component is then bootstrapped using the moving block bootstrap (mbb), and to every created bootstrap version of the remainder, the trend and seasonal components are added, and then the Box-Cox transformation is inverted. So a random pool of similar bootstrapped time

series is generated (in our case 100). After applying the forecasting method to each time series, the forecasts are aggregated by median. In Fig. 2 the results of mbb method applied on the Irish and Slovak data are shown. As it can be seen the method produces very noisy time series when the original data is also noisy (Slovak data). For this reason, we have proposed two new bootstrap methods for time series for better adaptation to noisy time series.

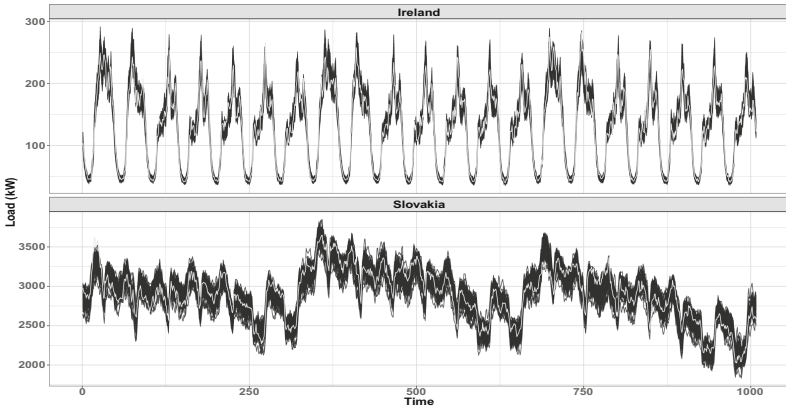


Fig. 2. Mbb method used on two different time series of length three weeks from Ireland and Slovakia. The original time series is illustrated with the light grey colour.

The first one of our newly proposed methods is a modification of the previous one (we will refer to as *smo.mbb*). This method is extended with one important procedure for ensuring the noise reduction. Before application of the moving block bootstrap method on remainder part of the time series, the remainder part is smoothed by simple exponential smoothing. The smoothing factor was set to 0.05 in our case. By this calculation, the remained part is significantly denoised, so the new bootstrapped time series do not fluctuate as in the original approach (mbb). The difference between mbb and *smo.mbb* method applied on Slovak data is shown in Fig. 3.

The second newly proposed bootstrap method for time series is clustering-based (we will refer to as *KM.boot*). It firstly uses K-means and automatic determination of the number of clusters, as described in Sect. 4.1, to cluster univariate time series to K groups (K was set to 12 – 20). Then new bootstrap time series are created from sampling from values of clusters. For example, if x_1 is value of the time series that belongs to second cluster, the new value of time series is randomly sampled value from the second cluster. For better illustration, the clustered time series by K-means and final bootstrapped time series by *KM.boot* method are shown in Fig. 4. We can see that bootstrapped time series have very low variance between each other, so *KM.boot* reduces noise dramatically.

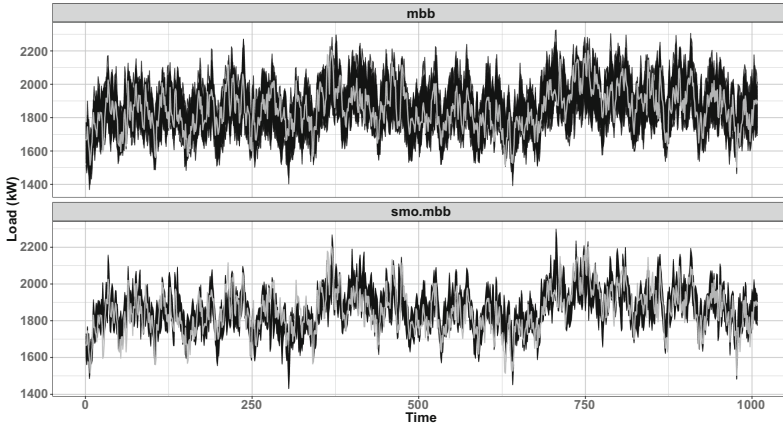


Fig. 3. The comparison of mbb and smo.mbb methods on time series of length three weeks from Slovakia. The original time series is illustrated with the light grey colour.

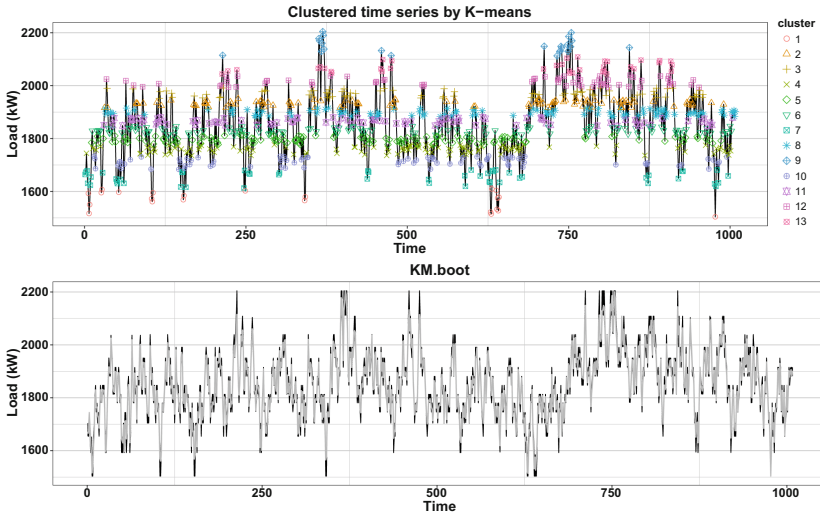


Fig. 4. The upper image shows clustered univariate time series to 13 clusters by K-means. The second image shows bootstrapped time series by KM.boot method. The original time series is illustrated with the light grey colour.

Ensemble Learning. We have implemented six different ensemble learning methods that we can divide into these three groups:

- (a) Simple aggregation based - average and median,
- (b) Naive cluster based - average of medians of methods,
- (c) Cluster based - K-means based, DBSCAN based and OPTICS based.

There are other widely used ensemble learning methods, which are error based, so they weight each prediction method by its performance. We proposed ensemble learning methods, which are structure based, so it uses unsupervised learning to create a final ensemble forecast. As we have found and experimentally proven, only unsupervised approaches are suitable for time series created by clustering, which are newly generated in each data window. Reason of this claim is that each created clustered time series needs to apply different forecasting method.

Average and median ensemble simply aggregates all available forecasts, in our case, there are $6 \times 100 = 600$ forecasts, which were produced by bagging.

The first cluster-based method uses a priori information on which forecasting method was used. Each method creates a median. After this, a set of medians (in this case having a set length of 6) is averaged to a final ensemble forecast.

The next three methods are cluster-based. Before using a clustering algorithm on a dataset of forecasts (matrix of dimension 600×48), the Principal Component Analysis is used to extract just the first three principal components in order to reduce noise. The K-means based procedure (Sect. 4.1) was used to create clusters of forecasts. First, DB-indexes were computed and an optimal number of clusters in the range of 3–8 was found. Next K-means produced clusters with corresponding centroids, which were averaged to the final ensemble forecast.

All ensemble methods mentioned above used all forecasts to produce the final one, even when anomalous, which could cause loss of forecasting accuracy. For this reason, density-based clustering methods which can automatically filter noisy objects were implemented. First of them, DBSCAN [19] (Density-Based Spatial Clustering of Applications with Noise) clustering algorithm was used. It requires two parameters: ϵ (set to 1.45) and *minPts* (set to 8). Ensemble forecast is created by the average of medians of clusters. The biggest drawback of this approach is that these parameters in the whole process of evaluation are set statically and not dynamically. However, deviations are reduced by principal components normalisation, which guarantees stability in the range of data values.

For producing density-based clustering that automatically adapts to the shape of objects, OPTICS [20] (Ordering Points To Identify the Clustering Structure) algorithm with automatic ξ -cluster procedure was implemented. ξ defines the degree of steepness (set to 0.045), which is applied in the so-called reachability plot of distances. The final ensemble forecast is the median of medians of clusters. The accuracy of the load forecast was measured by MAPE (Mean Absolute Percentage Error). MAPE is defined as $100 \times \frac{1}{n} \sum_{t=1}^n \frac{|x_t - \bar{x}_t|}{x_t}$, where x_t is a real consumption, \bar{x}_t is the forecasted load and n is a length of data.

5 Experiments

We have performed several experiments to evaluate our implemented methods. The Ireland testing dataset contains 3 months of measurements from the year 2010 (1.2.2010 – 28.2.2010, 1.5.2010 – 31.5.2010 and 1.8.2010 – 31.8.2010), comprising 90 days. The Slovak testing dataset contains also 3 months of measurements from years 2013 and 2014 (23.9.2013 – 26.10.2013, 10.2.2014 – 11.3.2014

and 2.6.2014 – 1.7.2014), comprising 94 days. Moreover we had additional data coming from 21 days before each of the six tested periods that were used in clustering and train forecasting methods. The source code of the all implemented methods is available online².

Table 1 summarises the results of basic methods used without bagging. Besides the comparison of average values of MAPE, the p-values of Wilcoxon rank sum test are also shown. They show whether forecast errors of the clustering approach have significantly lower values in comparison with the simple aggregation approach. Corresponding p-values show that in both datasets clustering of consumers significantly improves the accuracy of forecasting in four cases from six, but in the meaning of average values of MAPE, clustering improves the accuracy of forecasting in all cases. The CART forecasting method was the best basic method on Irish data and the EXP on Slovak data.

Table 1. Average daily MAPE (%) of 6 forecasting methods evaluated on two datasets and two types of aggregation. Agg. represents simple aggregation and Clust. clustering approach. Bold values represent lowest MAPE. P-values less than 0.05 are bold.

	Agg.-Irel.	Clust.-Irel.	p-value	Agg.-Slov.	Clust.-Slov.	p-value
CART	3.9101	3.8140	0.0270	3.1000	3.0761	0.0922
CTREE.lag	4.0828	3.8507	0.0009	3.0391	2.8969	<0.0001
CTREE.dft	4.1246	3.8757	<0.0001	3.1308	2.9854	<0.0001
STL+ARIMA	4.0718	3.8943	0.0248	2.7567	2.7186	0.0302
STL+EXP	4.2750	4.1866	0.5560	2.6887	2.6318	0.1644
EXP	4.8086	4.6508	0.1291	2.4130	2.3957	0.0152

The first comparison of ensemble learning forecasting methods is shown in Table 2, where for time series analysis methods mbb bootstrapping was used. CTREE.dft.bagg method was the best basic bagged method on Irish data and CTREE.lag.bagg on Slovak data. The best ensemble method on Irish data was the median method. On the simple aggregated Slovak data the average method had the lowest MAPE. The lowest MAPE on clustered Slovak dataset has been achieved by the OPTICS-based ensemble method. A significant improvement of results with the clustering approach on Irish data was in 5 of the 12 cases. On the other hand, clustering helped in 3 of the 12 cases on Slovak data. Notice that the best results by bagged basic methods are different from results using best basic methods from Table 1. In the case of Irish dataset, we improved results by 0.11% points of MAPE on the simple aggregated dataset and 0.08% points on the clustered dataset. However, in the case of Slovak dataset, result was worsen by 0.52% points of MAPE on the simple aggregated dataset and 0.45% points on the clustered dataset. It implies that mbb bagging method for time series analysis methods on Slovak dataset decreases accuracy of forecasts.

² <https://github.com/PetoLau/UnsupervisedEnsembles>.

Table 2. Average daily MAPE (%) of 12 forecasting methods evaluated on two datasets and two types of aggregation. Agg. represents simple aggregation of consumption and Clust. clustering approach. Bold values represent the lowest MAPE among bagged basic and among ensemble methods. P-values less than 0.05 are bold.

	Agg.-Irel.	Clust.-Irel.	p-value	Agg.-Slov.	Clust.-Slov.	p-value
CART.bagg	3.8630	3.7658	0.0177	3.0881	3.0598	0.1227
CTREE.lag.bagg	3.9708	3.8166	0.0004	2.9342	2.8506	0.0007
CTREE.dft.bagg	3.8040	3.7342	0.2382	3.0099	2.9557	0.0060
STL+ARIMA.mbb	3.9678	3.9025	0.0011	3.0188	3.0069	0.3531
STL+EXP.mbb	3.9804	4.0107	0.9296	3.0332	3.0020	0.1681
EXP.mbb	4.0931	4.0665	0.2070	2.9736	2.9322	0.0620
Average	3.8453	3.8019	0.0915	2.9424	2.9336	0.1991
Median	3.7768	3.7252	0.2744	2.9666	2.9400	0.1314
AveMedians	3.8125	3.7904	0.3437	2.9440	2.9217	0.0867
K-means	5.0296	4.1082	0.0003	3.0575	4.0662	0.4985
DBSCAN	4.1880	3.9231	0.0360	2.9700	2.9628	0.4061
OPTICS	3.8992	3.8243	0.1968	2.9921	2.8999	0.0164

Table 3 shows p-values of comparisons of Tables 1 and 2, where it is tested if the bagging basic forecasting methods improve the forecasting accuracy significantly. On the simple aggregated Irish data, all methods achieved statistically significant results, except the CART and the STL+ARIMA method. On the Slovak data, the bagging regression trees had significant results, but time series analysis methods had failed. This is caused by the mbb method that is not adaptable on noisy and fluctuated time series that is present in Slovak data.

As it was described in Sect. 4.2, for this reason, we proposed two new bootstrapping methods to overcome noisy character of time series. It was experimentally found the smo.boot bootstrapping method improves results on the Irish dataset and the KM.boot bootstrapping method improves results on the Slovak dataset. In the opposite scenario, these methods decrease accuracy of

Table 3. P-values of testing if bagging six basic forecasting methods improves forecasting accuracy significantly.

	Agg.-Irel.	Clust.-Irel.	Agg.-Slov.	Clust.-Slov.
CART	0.0592	0.0033	0.0916	0.0027
CTREE.lag	0.0004	0.0293	<0.0001	<0.0001
CTREE.dft	<0.0001	<0.0001	<0.0001	0.0012
STL+ARIMA.mbb	0.0646	0.0433	0.9854	0.9999
STL+EXP.mbb	0.0272	0.0184	0.9988	0.9999
EXP.mbb	0.0001	0.0003	1.0000	1.0000

forecast, so their results are not present in the tables. Table 4 shows these results, whereby on the Irish dataset the results achieved by smo.boot are showed and on the Slovak dataset the results achieved by KM.boot are showed. We can see that using the Irish dataset, the results of forecasts were improved dramatically by smo.boot bootstrapping method for time series analysis methods in the meaning of comparison of results presented in Tables 1 and 2. On the Slovak dataset, the results of forecasts were improved dramatically by KM.boot bootstrapping method for time series analysis methods in the meaning of comparison with the original mbb method (Table 2). However, they have still worse results compared to basic forecasting methods from Table 1. Clustering Irish data improved significantly forecasting results against the simple aggregated Irish data in 8 from 12 cases. Clustering Slovak data improved significantly forecasting results against simple aggregated Slovak data in 10 from 12 cases. The improvement of results achieved by clustering of consumers (better than in the case of usage mbb) is proof that our two new proposed bootstrap methods are working well.

Table 4. Average daily MAPE (%) of 12 forecasting methods evaluated on two datasets and two types of aggregation. Agg. represents simple aggregation of consumption and Clust. clustering approach. Bold values represent the lowest MAPE among bagged basic and among ensemble methods. P-values less than 0.05 are bold.

	Agg.-Irel.	Clust.-Irel.	p-value	Agg.-Slov.	Clust.-Slov.	p-value
boot. method	smo.mbb	smo.mbb		KM.boot	KM.boot	
CART.bagg	3.8630	3.7658	0.0177	3.0897	3.0598	0.1227
CTREE.lag.bagg	3.9708	3.8166	0.0004	2.9342	2.8506	0.0007
CTREE.dft.bagg	3.8040	3.7342	0.2382	3.0099	2.9557	0.0060
STL+ARIMA	3.8709	3.7151	0.0103	2.8452	2.7253	0.0023
STL+EXP	3.8541	3.6958	0.0078	2.8157	2.6543	<0.0001
EXP	3.9274	3.7573	0.0158	2.5631	2.4283	<0.0001
Average	3.7560	3.6788	0.2059	2.6765	2.6400	0.0256
Median	3.7402	3.6213	0.0662	2.7121	2.6717	0.0672
AveMedians	3.7341	3.6534	0.1753	2.6808	2.6406	0.0263
K-means	3.9516	3.6919	0.0042	2.6659	2.6132	0.0294
DBSCAN	3.8132	3.6589	0.0383	2.6472	2.5753	0.0261
OPTICS	3.8063	3.6345	0.0380	2.5679	2.6978	0.0060

Table 5 shows p-values of comparisons (differences) of Tables 1 and 4, where it is tested if the bagging basic time series analysis forecasting methods improves forecasting accuracy significantly. On the Irish dataset, now all methods achieved statistically significant improvement. On the Slovak data, the bagging time series analysis methods had again failed. We consider that bootstrapping methods for time series are not appropriate to use when time series shows strong double-seasonal pattern, trend changing and noisy character.

Table 5. P-values of testing if bagging three basic time series analysis forecasting methods improves forecasting accuracy significantly.

	Agg.-Irel.	Clust.-Irel.	Agg.-Slov.	Clust.-Slov.
boot. method	smo.mbb	smo.mbb	KM.boot	KM.boot
STL+ARIMA	0.0001	0.0004	0.9904	0.7514
STL+EXP	0.0001	<0.0001	0.9997	0.6691
EXP	<0.0001	<0.0001	1.0000	0.9503

The last analysis of results are shown in Table 6. The significance of the best ensemble approach against the best bagged method is shown, so results from Tables 2 and 4 are analysed. On the Irish data, two times from four cases the median ensemble method was significantly better than basic bagged method. On the Slovak data, ensembles were not significantly better nor in one case.

Table 6. P-values from hypothesis if ensemble method is better than forecasting method with bagging.

Agg.-Irel.	Clust.-Irel.	Agg.-Slov.	Clust.-Slov.
Ctree.dft-Median	Ctree.dft-Median	Ctree.lag-Average	Ctree.lag-Optics
0.0451	0.1563	0.6157	0.8579
smo.mbb	smo.mbb	KM.boot	KM.boot
Ctree.dft-AveMedians	STL+EXP-Median	EXP-Optics	EXP-DBSCAN
0.0989	<0.0001	0.2972	0.9890

6 Conclusion

In our paper we have proposed and tested two techniques for forecasting electricity load. We have compared and implemented various ensemble learning methods in order to compare its forecasting accuracy using aggregated and clustered electricity load. Six base forecasting methods combined with four different bagging (bootstrapping) methods and six unsupervised ensemble approaches, combining all available forecasts created by bootstrap methods, were evaluated. Two new methods for bootstrapping time series were proposed in order to handle noisy and fluctuate time series. One of them was a modification of mbb method that uses exponential smoothing to the noisy part of a time series. The second one was clustering-based method that uses K-means and sampling from created clusters. A new approach based on unsupervised ensemble learning in combination with clustered load was proposed and evaluated. The clustering of consumers was performed by efficient preprocessing using estimated regression coefficients as a representation of time series, K-means method and optimally finding a number

of clusters by DB-index. We have proven that the bagging of regression trees significantly improves accuracy on both aggregated and clustered load. On the other hand, bagging of time series analysis methods can be unreliable, because of weak adaptivity to noisy and fluctuated data. Our newly proposed bootstrapping methods improved results of forecasting against mbb method, but still results for Slovak dataset were not satisfying. Unsupervised ensemble learning approaches performed better than any other forecasting methods on Irish smart meter data. However, they failed on Slovak dataset because of instability of bootstrapping methods on fluctuate and noisy Slovak data.

For this reason we conclude that unsupervised ensemble learning is not suitable for every type of aggregated and also clustered load forecasting. However, the clustering of consumers itself stably improved forecasting accuracy of all methods. Smart meter data are often noisy and fluctuated. They force us to develop more robust methods to the detect trend shift (concept drift) and handle the noisy character of data. In future work, we want to focus on ensemble and clustering methods that are more adaptable for the aforementioned problems.

Acknowledgments. This work was partially supported by the Scientific Grant Agency of The Slovak Republic, Grant No. VG 1/0752/14 and STU Grant scheme for Support of Young Researchers.

References

1. Laurinec, P., Lucká, M.: Comparison of representations of time series for clustering smart meter data. In: Proceedings of WCECS, pp. 458–463 (2016)
2. Laurinec, P., Lóderer, M., Vrabecová, P., Lucká, M., Rozinajová, V., Ezzeddine, A.B.: Adaptive time series forecasting of energy consumption using optimized cluster analysis. In: Proceedings of IEEE ICDMW, pp. 398–405 (2016)
3. Adhikari, R., Verma, G., Khandelwal, I.: A model ranking based selective ensemble approach for time series forecasting. *Procedia Comput. Sci.* **48**, 14–21 (2015)
4. Shen, W., Babushkin, V., Aung, Z., Woon, W.L.: An ensemble model for day-ahead electricity demand time series forecasting. In: *e-Energy 2013*, pp. 51–62. ACM (2013)
5. Grmanová, G., Laurinec, P., Rozinajová, V., Ezzeddine, A.B., Lucká, M., Lacko, P., Vrabecová, P., Návrat, P.: Incremental ensemble learning for electricity load forecasting. *Acta Polytech. Hung.* **13**(2), 97–117 (2016)
6. Shahzadeh, A., Khosravi, A., Nahavandi, S.: Improving load forecast accuracy by clustering consumers using smart meter data. In: Proceedings of IJCNN (2015)
7. Wijaya, T.K., Vasirani, M., Humeau, S., Aberer, K.: Cluster-based aggregate forecasting for residential electricity demand using smart meter data. In: Proceedings of IEEE International Conference on Big Data, pp. 879–887 (2015)
8. Arthur, D., Vassilvitskii, S.: K-means++: the advantages of careful seeding. In: *SODA 2007*, pp. 1027–1035 (2007)
9. Davies, D.L., Bouldin, D.W.: A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **2**, 224–227 (1979)
10. Cleveland, R.B., et al.: Seasonal-trend decomposition procedure based on LOESS. *J. Off. Stat.* **6**, 3–73 (1990)

11. Box, G.E.P., Jenkins, G.M.: *Time Series Analysis: Forecasting and Control*. Holden-Day, San Francisco (1970)
12. Holt, C.C.: Forecasting seasonals and trends by exponentially weighted moving averages, vol. 52. ONR Research Memorandum, Carnegie Inst. of Tech. (1957)
13. Hyndman, R.J., Koehler, A.B., Snyder, R.D., Grose, S.: A state space framework for automatic forecasting using exponential smoothing methods. *Int. J. Forecast.* **18**(3), 439–454 (2002)
14. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: *Classification and Regression Trees*. Chapman and Hall/CRC, Wadsworth (1984)
15. Hyndman, R.J., Khandakar, Y.: Automatic time series forecasting: the forecast package for R. *J. Stat. Softw.* **27**(3), 1–22 (2008)
16. Strasser, H., Weber, C.: On the asymptotic theory of permutation statistics. *Math. Methods Stat.* **8**, 220–250 (1999)
17. Breiman, L.: Bagging predictors. *Mach. Learn.* **24**(2), 123–140 (1996)
18. Bergmeir, C., Hyndman, R.J., Benítez, J.M.: Bagging exponential smoothing methods using STL decomposition and Box-Cox transformation. *Int. J. Forecast.* **32**(2), 303–312 (2016)
19. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the KDD*, pp. 226–231 (1996)
20. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: ordering points to identify the clustering structure. In: *Proceedings of ACM SIGMOD*, pp. 49–60 (1999)



Phenotype Prediction with Semi-supervised Classification Trees

Jurica Levatić^{1,2(✉)}, Maria Brbić³, Tomaž Stepišnik Perdih^{1,2}, Dragi Kocev^{1,2},
Vedrana Vidulin^{1,3,4}, Tomislav Šmuc³, Fran Supek^{3,5}, and Sašo Džeroski^{1,2}

¹ Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
jurica.levatic@ijs.si

² Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

³ Division of Electronics, Ruder Boskovic Institute, Zagreb, Croatia

⁴ Faculty of Information Studies, Novo Mesto, Slovenia

⁵ Center for Genomic Regulation, Barcelona, Spain

Abstract. In this work, we address the task of phenotypic traits prediction using methods for semi-supervised learning. More specifically, we propose to use supervised and semi-supervised classification trees as well as supervised and semi-supervised random forests of classification trees. We consider 114 datasets for different phenotypic traits referring to 997 microbial species. These datasets present a challenge for the existing machine learning methods: they are not labelled/annotated entirely and their distribution is typically imbalanced. We investigate whether approaching the task of phenotype prediction as a semi-supervised learning task can yield improved predictive performance. The results suggest that the semi-supervised methodology considered here is especially helpful when using single trees, especially when the amount of labeled data ranges from 20 to 40%. Similar improvements can be seen when the presence of the phenotype is very imbalanced.

Keywords: Semi-supervised learning · Phenotype · Decision trees
Predictive clustering trees · Random forests · Binary classification

1 Introduction

The most common task in machine learning is supervised learning, where the goal is to predict the value of a target attribute of an example by using the values of descriptive attributes. Supervised methods often need a large amount of labeled data to learn a predictive model with a satisfying predictive performance. However, in many real-life problems, such as phonetic annotation of human speech, protein 3D structure prediction, and spam filtering, only a few labeled examples are available to learn from because of the expensive and/or time-consuming annotation procedures. Contrary to labeled examples, unlabeled examples are often freely available in vast amounts. For example, human speech can be recorded from radio broadcasts, while DNA sequences of proteins can

be extracted from gene databases. Semi-supervised learning (SSL) emerged as an answer to the problem of labeled data scarcity [1], with an idea to exploit freely/easily available unlabeled examples to get better predictive performance than the one achieved using labeled data alone.

In this work, we are concerned with the task of microbial phenotype prediction. Phenotypes are defined as variations in observable characteristics of an organism. Microbial organisms display a large diversity of possible phenotypic traits, such as ability to inhabit different environments, adaptation to extreme conditions and association to different hosts. The annotation of organisms with phenotypes is important for understanding the genetic basis of phenotypes. It often requires expensive experimental measurements and time-consuming manual curation, hence there is a huge amount of unlabeled organisms. On the other hand, phenotypes can be efficiently predicted from genome [2–5] and metagenome data [6].

Thanks to the emergence of DNA sequencing technology, the number of sequenced genomes is rapidly increasing, making unlabeled data easily available. This makes the problem of phenotype prediction well suited for semi-supervised learning. In this work, we explore whether better predictive performance can be achieved with semi-supervised machine learning methods than with supervised methods that have been used for this task in [7]¹, namely classification trees and random forests. To the best of our knowledge, this is the first application of semi-supervised learning for microbial phenotype prediction.

In this work, we compare the predictive performance of supervised and semi-supervised classification trees and random forests thereof [8] to predict 114 phenotypes of 997 microbial organisms. These datasets pose interesting challenges for existing machine learning methods because the annotations are not complete and the available datasets are imbalanced. To this end, we investigate whether we can benefit from using semi-supervised learning under these difficult conditions. In a nutshell, the results reveal that the semi-supervised classification trees can improve the predictive performance over supervised classification trees in cases where the amount of labeled data is in the range 20–40% and for phenotypic traits that are not extremely rare.

The rest of this paper is organized as follows. Section 2 describes the semi-supervised methods used in this study, while Sect. 3 describes the data used for phenotype prediction. Section 4 specifies the experimental design. The results of the empirical investigations are presented and discussed in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Methods

In this work, we consider semi-supervised classification trees and semi-supervised random forests [8], which are based on the predictive clustering trees (PCTs) [9] and ensembles thereof [10]. PCTs view a decision tree as a hierarchy of clusters,

¹ Phenotype predictions from [7] are available at protraits.irb.hr.

where the top-node corresponds to one cluster containing all the data. This cluster is then recursively partitioned into smaller clusters while moving down the tree. Semi-supervised PCTs are implemented in the CLUS system [11] (implementation available at <http://kt.ijs.si/jurica.levatic/>). In this section, we briefly describe semi-supervised trees and random forests, while for more details we refer the reader to the work of Levatić et al. [8].

Supervised classification trees evaluate the quality of splits on the basis of the class labels, by using, for example information gain or gini impurity as a quality measure. Consequently, the resulting clusters (i.e., groups of examples defined by splits in the tree) are homogeneous only with respect to the class label. Semi-supervised PCTs [8], on the other hand, measure the quality of splits considering both the class labels and descriptive attributes. Therefore, the resulting clusters are homogeneous with respect to both the descriptive attributes and the class labels. Note that, only the descriptive attributes are known for unlabeled examples, thus, such semi-supervised trees can exploit them during the tree construction - contrary to supervised trees. The rationale behind the described semi-supervised classification trees is the semi-supervised cluster assumption [1]: *If examples are in the same cluster, then they are likely of the same class.*

The semi-supervised PCTs are based on the standard *top-down induction of decision trees* (TDIDT) algorithm (see Table 1), which takes as input a set of examples E and outputs a tree. The heuristic score (h) that is used for selecting the tests (t) to put in the internal tree nodes is reduction of impurity caused by partitioning (\mathcal{P} , Table 1, line 3 of the *BestTest* procedure) the examples according to the tests.

In supervised PCTs, the impurity for each set of examples E is calculated as the gini impurity (Table 1, line 5 of the *BestTest* procedure):

$$\text{Impurity}(E) = \text{Gini}(E, Y). \quad (1)$$

Table 1. The top-down induction algorithm for decision trees construction.

procedure InduceTree	procedure BestTest
Input: A dataset E	Input: A dataset E
Output: A predictive clustering tree	Output: the best test (t^*), its heuristic score (h^*) and the partition (\mathcal{P}^*) it induces on the dataset (E)
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$	1: $(t^*, h^*, \mathcal{P}^*) = (\text{none}, 0, \emptyset)$
2: if $t^* \neq \text{none}$ then	2: for each possible test t do
3: for each $E_i \in \mathcal{P}^*$ do	3: $\mathcal{P} =$ partition induced by t on E
4: $\text{tree}_i =$	4: $h = \text{Impurity}(E) -$
5: InduceTree(E_i)	5: $\sum_{E_i \in \mathcal{P}} \frac{ E_i }{ E } \text{Impurity}(E_i)$
6: return	6: if $(h > h^*) \wedge \text{Acceptable}(t, \mathcal{P})$ then
7: node(t^* , $\bigcup_i \{\text{tree}_i\}$)	7: $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
8: else	8: return $(t^*, h^*, \mathcal{P}^*)$
9: return	
10: leaf(Prototype(E))	

As mentioned before, to identify the best splits, the impurity function of semi-supervised PCTs takes into account both the target attribute (i.e., the class labels) and the descriptive attributes. This is achieved by changing the equation for the calculation of impurity for supervised PCTs (Eq. 1). Impurity of a set of examples E (which may contain labeled and unlabeled examples) is calculated as a weighted sum of impurities over the target attribute (Y) and impurities over the descriptive attributes (X_i):

$$Impurity_{SSL}(E) = w \cdot Impurity(E_l, Y) + \frac{1-w}{D} \cdot \sum_{i=1}^D Impurity(E, X_i), \quad (2)$$

where $E = E_l \cup E_u$ is the dataset available at a node of the tree, D is the number of descriptive attributes, X_i is the i^{th} descriptive attribute, and $w \in [0, 1]$ is a weight parameter.

The impurity of the target attribute Y is calculated as gini impurity over a set of labeled examples E_l . Differently from the target attribute, which is nominal, the descriptive attributes can be either nominal or numeric, therefore, the two cases are considered separately: if the attribute is nominal as a measure of impurity gini impurity is used, whereas, if the attribute is numeric, as a measure of impurity variance is used.

The weight parameter w in (2) controls how much the target side or the descriptive side contribute to the calculation of the impurity. Consequently, this controls how much the unlabeled examples affect the learning of semi-supervised PCTs. Namely, depending on the values of the w parameter, semi-supervised PCTs can range from fully supervised trees (i.e., $w = 1$) to completely unsupervised trees (i.e., $w = 0$). This aspect is important since unlabeled examples can sometimes cause semi-supervised algorithms to perform worse than their supervised counterparts [12–14]. The w parameter acts as a safety mechanism of semi-supervised PCTs, enabling them to control the influence of unlabeled examples and adapt to a given dataset.

If no acceptable test is found because some stopping criteria is met (e.g., minimum number of examples in the leaf has reached the user predefined value, the variance reduction is not relevant etc.) then the algorithm places a leaf node at that position. In each leaf node, the prototype for the examples belonging to that leaf node is calculated (by using the function $Prototype(E)$ from the *InduceTree* procedure in Table 1 at line 10) and stored.

By using semi-supervised PCTs, it is possible to build semi-supervised random forests. A random forest [15] is an ensemble of trees, where diversity among the trees is obtained by making bootstrap replicates of the training set, and additionally by randomly selecting the subset of descriptive attributes used to evaluate the splits. Random forests often substantially improve the predictive performance of single trees, however, the interpretability aspect of trees is lost. Semi-supervised random forests of PCTs are built by using semi-supervised PCTs as members of the ensemble, instead of using supervised PCTs. In semi-supervised random forests, the bootstrap sampling procedure is modified to perform

stratified bootstrap sampling (considering the proportions of labeled and unlabeled examples) to avoid having bootstrap samples consisting only of unlabeled examples.

We next analyze the computational complexity of semi-supervised PCTs. We first recall the procedures that contribute to the computational complexity of *supervised* PCTs. These are as follows: sorting the values of D descriptive attributes ($O(DN \log N)$), calculating the best split for T target variables ($O(TDN)$), and applying the split to the N (labeled) training examples ($O(N)$). Assuming that the depth of the tree is in the order of $O(\log N)$ [16], the total computational complexity of constructing a single PCT is $O(DN \log^2 N) + O(TDN \log N) + O(N \log N)$.

We then consider what changes from supervised PCTs to semi-supervised PCTs. This is, first, the value of N : In the case of semi-supervised PCTs, the number of training examples is equal to the number of labeled and unlabeled examples combined, i.e., $N = N_l + N_u$, instead of $N = N_l$. Second, SSL-PCTs consider both D descriptive attributes and T target variables when the split is calculated, thus the complexity of this step is $O((T + D)DN)$. The total computational complexity of learning a single SSL-PCT is thus $O(DN \log^2 N) + O((T + D)DN \log N) + O(N \log N)$. This cost is then linearly extended to random forests of PCTs similarly as in [10]. Additionally, one should also consider the cost for obtaining the optimal value for the w parameter, which is usually performed using an inner cross-validation procedure.

3 Data Description

Prokaryotic genome sequences and gene annotations were downloaded from the NCBI Genomes database and COG/NOG gene families were downloaded from eggNOG 3 [17]. In our analysis, we considered species that have a genome quality score greater or equal to 0.9 (out of 1) [22]. Higher score corresponds to the higher level of completeness of sequenced genome data, where scores of 0.8 or higher indicate that a genome can be safely used for standard comparative genomics analysis. Phenotype annotations are NCBI+Bacmap labels as in [7], collected from the NCBI microbial genome projects list ('lproks0' table) and from the BacMap database [18], in total 114 different phenotypic traits. We considered only species having at least one assigned phenotype label, resulting in 997 species. Each example corresponds to one species labeled with a set of available phenotypic traits. For each species, the labels correspond to presence or absence of traits, thus, the task of phenotype prediction corresponds to a binary classification problem.

The labelling is, however, not exhaustive: For most of the phenotypes, only 30% of species are labeled (Fig. 1a). Hence, the dataset at hand contains unlabeled data, which can be exploited with semi-supervised methods. The class distribution of most of the phenotypes is unbalanced (Fig. 1b): Many traits appear at less than 10% of species, e.g. radiation-resistance phenotype and ability to withstand extremely high (hyperthermophilic organisms) or extremely low temperature (psychrophilic organisms).

In all experiments, we used the gene repertoire representation [2]. The features describing the species were encoded as the presence/absence of the clusters of orthologous (COG) and non-supervised orthologous (NOG) groups of proteins, resulting in the 80576 binary valued features. In order to reduce the dimensionality of the feature set we applied principal component analysis (PCA) as a preprocessing step and retained principal components explaining 90% of the variance. This resulted in 526 features, i.e., principal components.

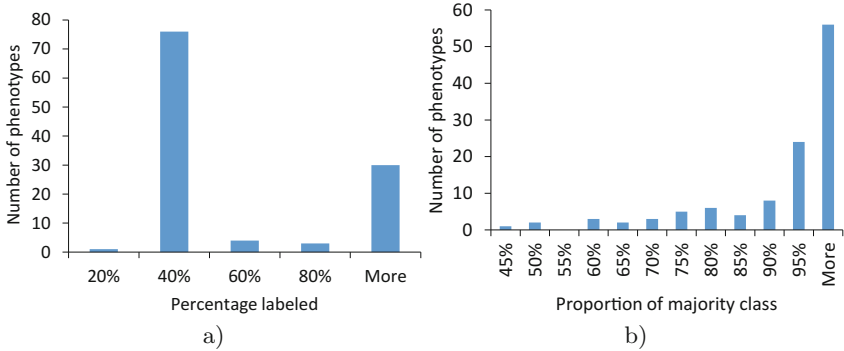


Fig. 1. (a) Histogram of the amount of labeled (relative to unlabeled) examples for each phenotype. (b) Histogram of the majority class distributions of phenotypes.

4 Experimental Design

We learn a separate model for each phenotype, transforming the problem of phenotype prediction into 114 binary classification tasks. We then approach these tasks with two learning paradigms: supervised and semi-supervised learning. In other words, we learn predictive models in the form of supervised classification trees (PCTs) and semi-supervised classification trees (SSL-PCTs) as well as supervised random forests and semi-supervised random forests. Performance was estimated with 10-fold cross validation procedure. The predictive performance reported in the results is the average of the performance values obtained from the 10 folds.

In the experiments, both supervised and semi-supervised trees are pruned with the procedure used in C4.5 classification trees [19]. The weight parameter w of semi-supervised algorithms was chosen from the set $\{0, 0.1, \dots, 0.9, 1\}$ by using internal 3-fold cross validation on the labeled part of the training set. We construct random forests consisting of 100 trees. The trees in random forests are not pruned and the number of random features at each internal node is set to the square root of the number of features, which in our case amounted to 23.

Next, we compare the performance of semi-supervised PCTs and semi-supervised random forests to their supervised counterparts. For every phenotype, examples with unknown labels were used as unlabeled data for learning the semi-supervised PCTs and ensembles thereof.

Furthermore, we investigate the influence of the amount of annotated phenotypes on the performance of the semi-supervised methods. More specifically, we analyze the performance of the predictive models across the different percentages of annotated phenotypes. Moreover, we juxtapose this influence with the influence of the imbalance of the class labels.

We also investigate how the value of the w parameter affects semi-supervised methods. To do this, we randomly select 4 phenotypes and learn semi-supervised models (both single PCTs and random forests) to predict them for all w in $\{0, 0.1, \dots, 0.9, 1\}$ the resulting performances. Additionally, we analyze the performance of predictive models for different values of the w parameter selected by the internal cross validation.

Finally, in our evaluation scenario we use truly unlabeled data, and not unlabeled data that is obtained by removing the labels as it is usually done in most SSL studies. Therefore, for each phenotype we use all the available unlabeled data. We have performed an analysis of the influence of the amount of the unlabeled data in [8]. The study revealed that the advantage of semi-supervised classification trees over supervised trees is dependant more on the dataset at hand, rather than on the amount of unlabeled data used, i.e., if the SSL algorithm wins, it is likely to win for different amounts of unlabeled data (on that dataset).

5 Results and Discussion

5.1 Predictive Performance

The performances of predicting 114 microbial phenotypic traits with supervised and semi-supervised trees and random forests are presented in Fig. 2. Because class distribution was very imbalanced for some phenotypes, we used F1 score (harmonic mean of precision and recall) in addition to accuracy to measure the performance. We can observe that for many of the traits, semi-supervised algorithms outperform their supervised counterparts, suggesting that semi-supervised methods can successfully exploit unlabeled data and more accurately predict microbial phenotypes. The advantage of semi-supervised methods is, however, not observed for all phenotypes. This is expected, since several researchers found that the success of semi-supervised methods is, in general, dataset dependent [20]. In other words, it cannot be expected that semi-supervised methods will win against supervised ones for all cases. Furthermore, several researchers have found that semi-supervised learning may sometimes perform worse than supervised learning [12–14]. The numbers of wins, ties and losses of semi-supervised algorithms compared to their supervised counterparts in accuracy and F1 score can be seen in Table 2. Ties were results where the difference in performance was smaller than 0.01.

Our results also suggest that improving (with unlabeled data) a supervised random forest is a harder task than improving over a supervised tree: The number of wins of semi-supervised random forests is lower than the number of wins of semi-supervised PCTs. This observation complies with previous findings [8].

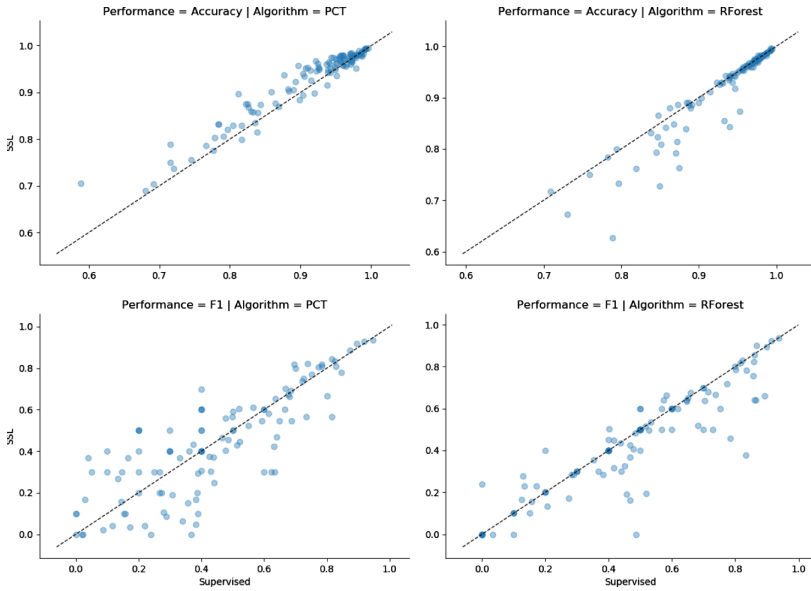


Fig. 2. Each dot represents the performance on one phenotype of supervised and semi-supervised methods. Values above the diagonal (dashed line) denote that the semi-supervised algorithm performed better. Darker color means greater density of dots. (Color figure online)

Table 2. Numbers of wins, ties and losses of semi-supervised algorithms compared to their supervised counterparts.

	PCT:Acc	PCT:F1	RForest:Acc	RForest:F1
Wins	62	50	3	16
Ties	44	15	91	58
Losses	8	49	20	40

We consider that this is due to the fact that ensembles are very powerful predictive models, which are able to exploit all the information in a given (labeled) dataset and approach the learnability borders of a given domain closer than a single predictive model. Thus, arguably, random forests do not benefit so much from additional information that unlabeled data bring, as compared to single trees.

We further analyze the results with the goal to identify phenotypes that are suitable for prediction with semi-supervised methods. The amount of available labeled data (relative to unlabeled) is an important factor for the performance of semi-supervised methods [8]. We therefore analyze the results from that aspect (Fig. 3). We can observe that semi-supervised single trees perform better with smaller amounts of labeled data according to accuracy and F1 score,

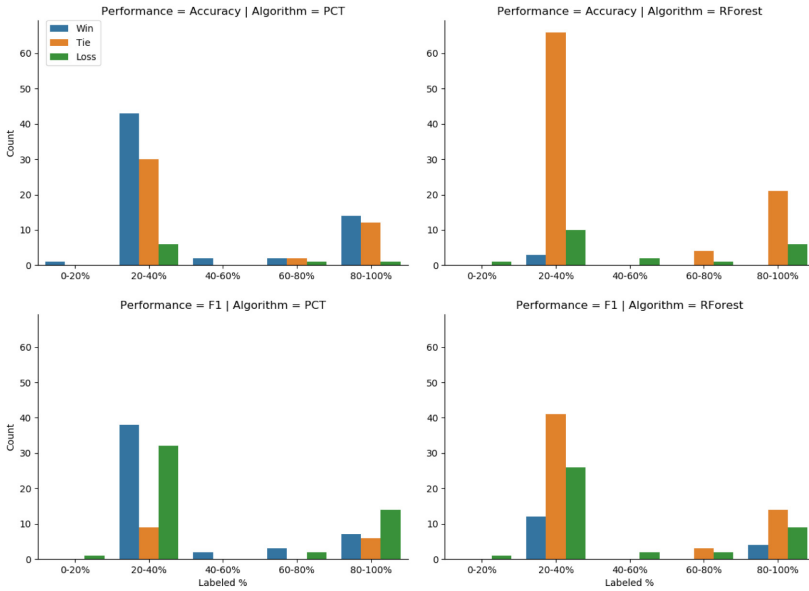


Fig. 3. The numbers of wins, ties and losses of semi-supervised PCTs and random forests versus their supervised counterparts, achieved for phenotypes with different amounts of labeled examples.

while supervised single trees have better F1 scores on phenotypes with a lot of labeled examples (over 80%). Supervised and semi-supervised Random forests are mostly tied, especially in terms of accuracy, however losses are more common than wins.

Recall that many of the phenotypes have very unbalanced classes (Fig. 1b). We next analyze whether the imbalance of the classes affects the performance of semi-supervised methods (Fig. 4). Interestingly, we can see that the semi-supervised methods achieve most wins in F1 score on phenotypes with the highest class imbalance. This holds for both single trees, where losses are more common than wins when the proportion of the majority class is less than 95%, and random forests, where the number of wins on the most imbalanced targets is almost the same as the number of losses, even though losses are far more common overall. Less surprisingly, we can also see that the vast majority of ties comes from phenotypes with the highest class imbalance.

5.2 Influence of the w Parameter

Figure 5 shows the accuracy of semi-supervised methods with different values of the w parameter for 4 randomly selected targets (phenotypes). We can see that in some cases its influence is minimal (e.g., random forests on target 2) but more often we need to select the right value for w to improve the performance over supervised methods. The best performance is achieved with different values

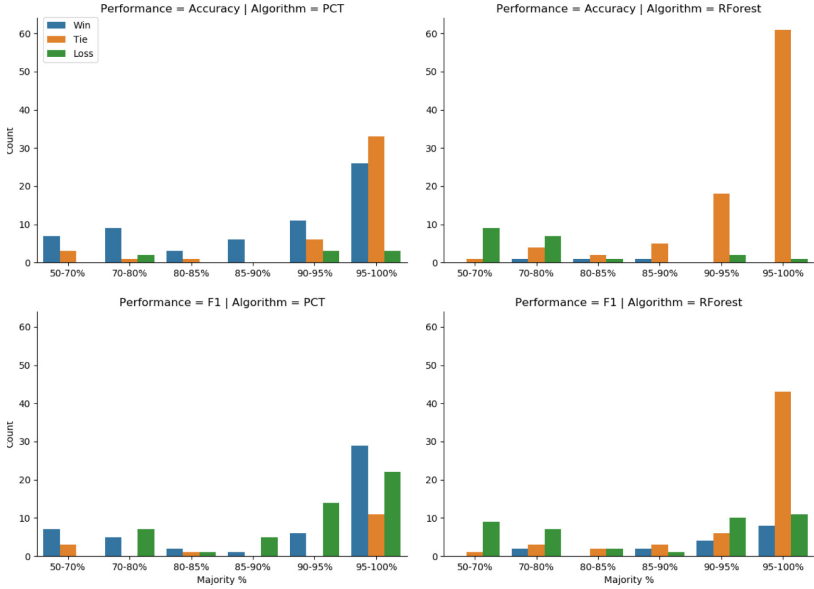


Fig. 4. The numbers of wins, ties and losses of semi-supervised PCTs and random forests versus their supervised counterparts, achieved for phenotypes with different proportion of the majority class.

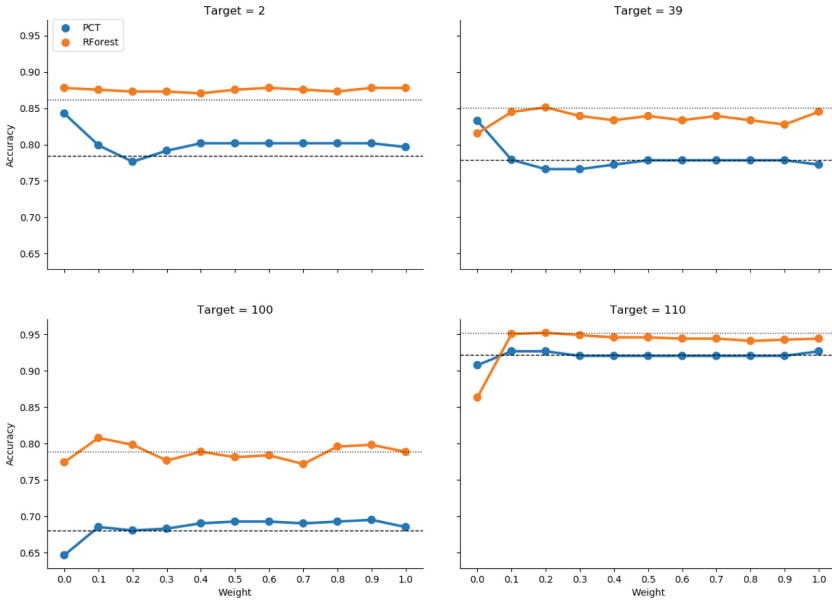


Fig. 5. The performance of semi-supervised methods for various values of w for 4 random phenotypes. The dashed lines represents the performance of supervised PCTs while the dotted line shows the performance of supervised random forests.

of the parameter, indicating that it should be tuned for every dataset. This is consistent with previous results on this topic [8] and the reason why we used internal cross validation to select it.

We also look at the numbers of wins, ties and losses according to the w selected (Fig. 6). Because the performance was measured with 10-fold cross validation and a different w was selected for each fold, we here compare the performances on each fold and not aggregated as before.

First, we note that for single trees $w = 0$ and $w = 1$ were the most common selections. Interestingly, when $w = 0$ is selected, accuracy is improved in most cases while the F1 score is close to even. Ties are most common for $w = 1$, which is to be expected. For random forests $w = 1$ is selected almost always, which contributes to the high number of ties in performance observed in the results previously.

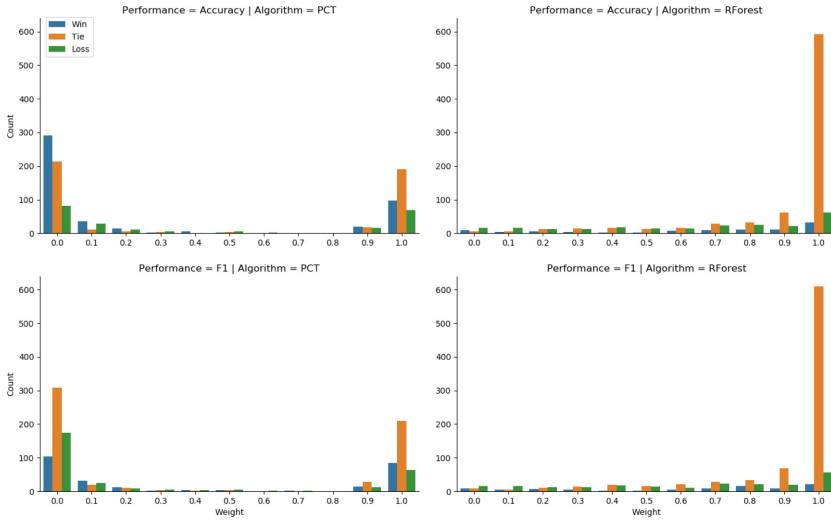


Fig. 6. Numbers of wins, ties and losses for different values of the w parameter selected by the internal cross validation.

6 Conclusions

In this work, we approach the task of phenotypic traits prediction using methods for semi-supervised learning. This task is important to understand the genetic basis for appearance of specific phenotypes. More specifically, we consider 114 datasets with different phenotypic traits referring to 997 microbial species. The datasets are not completely labelled and different amount of annotation is available for the different traits.

We investigate whether approaching the task of phenotype prediction as a semi-supervised learning task can yield improved predictive performance.

More specifically, we learn supervised and semi-supervised classification trees as well as supervised and semi-supervised random forests of classification trees. We then compare the performance of predictive models learned using supervised and semi-supervised methods.

The result suggest that the semi-supervised methodology considered here improves the accuracy of single trees and also their F1 score when the amount of labeled data ranges from 20 to 40%. Similar improvement can be seen when the presence of a phenotype is very imbalanced (proportion of the majority class over 95%). Improvement of random forests was rarer but also more common on previously mentioned groups of phenotypes. In applications where interpretable models are needed, semi-supervised classification trees should be favored over the supervised classification trees. We also showed that the performance of semi-supervised methods is sensitive to the value of the w parameter and that it should be tuned to each dataset.

We plan to further extend this work along several dimensions. To begin with, we plan to use phenotypes from other sources, specifically phenotypes from GOLD database [21] and especially biochemical phenotypes from [7] where the labeled examples are extremely scarce. Furthermore, we plan to consider other feature spaces, namely the proteome composition, gene neighborhoods and translation efficiency representations [7]. Next, we will compare the approaches presented here with other methods used for phenotype prediction including, but not limited to, SVMs and semi-supervised SVMs. Note that, considering the number of datasets considered here, such experiments will require massive computational power. Finally, we can treat the problem as a multi-label classification problem and obtain a partially labelled dataset that can be then approached from this perspective.

Acknowledgments. We acknowledge the financial support of the Slovenian Research Agency, via the grant P2-0103 and a young researcher grant to TSP, Croatian Science Foundation grants HRZZ-9623 (DescriptiveInduction), as well as the European Commission, via the grants ICT-2013-612944 MAESTRA and ICT-2013-604102 HBP. We would also like to acknowledge the joint support of the Republic of Slovenia and the European Union under the European Regional Development Fund (grant “Raziskovalci-2.0-FIŠ-52900”, implementation of the operation no. C3330-17-529008).

References

1. Chapelle, O., Schölkopf, B., Zien, A.: Semi-supervised Learning, vol. 2. MIT Press, Cambridge (2006)
2. MacDonald, N.J., Beiko, R.G.: Efficient learning of microbial genotype-phenotype association rules. *Bioinformatics* **26**(15), 1834 (2010)
3. Smole, Z., Nikolic, N., Supek, F., Šmuc, T., Sbalzarini, I.F., Krisko, A.: Proteome sequence features carry signatures of the environmental niche of prokaryotes. *BMC Evol. Biol.* **11**(1), 26 (2011)
4. Feldbauer, R., Schulz, F., Horn, M., Rattei, T.: Prediction of microbial phenotypes based on comparative genomics. *BMC Bioinform.* **16**(14), S1 (2015)

5. Brbić, M., Warnecke, T., Kriško, A., Supek, F.: Global shifts in genome and proteome composition are very tightly coupled. *Genome Biol. Evol.* **7**(6), 1519 (2015)
6. Chaffron, S., Rehrauer, H., Pernthaler, J., von Mering, C.: A global network of coexisting microbes from environmental and whole-genome sequence data. *Genome Res.* **20**(7), 947–959 (2010)
7. Brbić, M., Piškorec, M., Vidulin, V., Kriško, A., Šmuc, T., Supek, F.: The landscape of microbial phenotypic traits and associated genes. *Nucleic Acids Res.* **44**(21), 10074 (2016)
8. Levatić, J., Ceci, M., Kocev, D., Džeroski, S.: Semi-supervised classification trees. *J. Intell. Inf. Syst.* **49**(3), 461–486 (2017)
9. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: *Proceedings of the 15th International Conference on Machine learning*, pp. 55–63 (1998)
10. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. *Pattern Recogn.* **46**(3), 817–833 (2013)
11. Blockeel, H., Struyf, J.: Efficient algorithms for decision tree cross-validation. *J. Mach. Learn. Res.* **3**, 621–650 (2002)
12. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. *Mach. Learn.* **39**(2–3), 103–134 (2000)
13. Cozman, F., Cohen, I., Cirelo, M.: Unlabeled data can degrade classification performance of generative classifiers. In: *Proceedings of the 15th International Florida Artificial Intelligence Research Society Conference*, pp. 327–331 (2002)
14. Guo, Y., Niu, X., Zhang, H.: An extensive empirical study on semi-supervised learning. In: *Proceedings of the 10th International Conference on Data Mining*, pp. 186–195 (2010)
15. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
16. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Cambridge (2005)
17. Powell, S., Szklarczyk, D., Trachana, K., Roth, A., Kuhn, M., Muller, J., Arnold, R., Rattei, T., Letunic, I., Doerks, T., Jensen, L.J., von Mering, C., Bork, P.: eggNOG v3.0: orthologous groups covering 1133 organisms at 41 different taxonomic ranges. *Nucleic Acids Res.* **40**(D1), D284 (2012)
18. Stothard, P., Van Domselaar, G., Shrivastava, S., Guo, A., O’Neill, B., Cruz, J., Ellison, M., Wishart, D.S.: BacMap: an interactive picture atlas of annotated bacterial genomes. *Nucleic Acids Res.* **33**(suppl. 1), D317–D320 (2005)
19. Quinlan, J.R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco (1993)
20. Chawla, N., Karakoulas, G.: Learning from labeled and unlabeled data: an empirical study across techniques and domains. *J. Artif. Intell. Res.* **23**(1), 331–366 (2005)
21. Reddy, T., Thomas, A.D., Stamatis, D., Bertsch, J., Isbandi, M., Jansson, J., Mallajosyula, J., Pagani, I., Lobos, E.A., Kyripides, N.C.: The genomes online database (GOLD) v.5: a metadata management system based on a four level (meta)genome project classification. *Nucleic Acids Res.* **43**(D1), D1099 (2015)
22. Land, M.L., Hyatt, D., Jun, S.R., Kora, G.H., Hauser, L.J., Lukjancenko, O., Ussery, D.W.: Quality scores for 32,000 genomes. *Stand. genomic sci.* **9**(1), 20 (2014)



Structuring the Output Space in Multi-label Classification by Using Feature Ranking

Stevanche Nikoloski^{2,3(✉)}, Dragi Kocev^{1,2}, and Sašo Džeroski^{1,2}

¹ Department of Knowledge Technologies, Jožef Stefan Institute,
Ljubljana, Slovenia

{dragi.kocev,saso.dzeroski}@ijs.si

² Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
stevanche.nikoloski@ijs.si

³ Teagasc, Environment Soils and Land-Use Department,
County Wexford, Ireland

Abstract. Motivated by the increasing interest for the task of multi-label classification (MLC) in recent years, in this study we investigate a new approach for decomposition of the output space with the goal to improve the predictive performance. Namely, the structuring of the output/label space is performed by constructing a label hierarchy and then approaching the MLC task as a task of hierarchical multi-label classification (HMLC). Our approach is as follows. We first perform feature ranking for each of the labels separately and then represent each of the labels with its corresponding feature ranking. The construction of the hierarchy is performed by the (hierarchical) clustering of the feature rankings. To this end, we employ four clustering methods: agglomerative clustering with single linkage, agglomerative clustering with complete linkage, balanced k-means and predictive clustering trees. We then use predictive clustering trees to estimate the influence of the constructed hierarchies, i.e., we compare the predictive performance of models without exploiting the hierarchy and models using hierarchies constructed using label co-occurrences or per label feature rankings. Moreover, we investigate the influence of the hierarchy in the context of single models and ensembles of models. We evaluate the proposed approach across 8 datasets. The results show that the proposed method can yield predictive performance boost across several evaluation measures.

Keywords: Multi-label classification · Hierarchy construction
Feature ranking · Structuring of the label space

1 Introduction

Nowadays, the number of new applications of multi-label learning is steadily increasing, hence, the researchers are very interested to develop novel methods and new ideas related to multi-label classification and structuring of the

label/output space. Multi-label classification (MLC) is the task of learning from data examples where each example can be associated with multiple labels. MLC deals with a label dependencies and relations which is orthogonal with existing traditional methods which take into account label independence and learn independent functions from mapping from input space to the output (label) space. The different application problems include video and image annotations (new movie clips, genres), predicting genes and proteins (functional genomics), classification of a tweets and music into emotions, text classification (web-pages, bookmarks, e-mails,...) and others.

The MLC task is typically approached either by decomposing the MLC problem into multiple single class classification problems (i.e., problem transformation methods) or by modifying the algorithms to consider the multiple classes jointly (i.e., algorithm adaptation methods) [12]. In an extensive experimental study Madjarov et al. [7] show that the landscape of MLC methods is not simple: on some datasets problem transformation methods achieve top performance while on other datasets the algorithm adaptation methods are top performing. Furthermore, the study recommends the use of two algorithms for benchmarking: RF-PCT (Random forests of predictive clustering trees, an algorithm adaptation method) [5] and HOMER (Hierarchy Of Multi-label learnERs, a problem transformation method) [13]. The latter divides the label space into subspaces and then constructs classifiers for each of the subspace (e.g., label power set classifiers). This hints that the best performance might be obtained in between the spectrum of the algorithm adaptation and problem transformation methods. In other words, state-of-the-art MLC performance might be obtained by transforming the original MLC problem into several MLC problems and then learn predictive models (preferably using algorithm adaptation methods).

A crucial step in developing methods for output decomposition for MLC is the creation of the subspaces. More specifically, the goal is to find a dependency structure and consider jointly the labels that are inter-dependent. The construction of the output structure of the labels can be very tedious and expensive process, especially if domain experts are needed to complete the task. Moreover, selection of the representation language of the dependencies can be complicated task on its own. Typically, these dependencies are represented as hierarchies of labels [6]. The hierarchies can then be constructed in a data-driven manner using the descriptive space and/or the label space. This presents automatic and relatively efficient process to obtain the representation of the potential dependencies in the label space.

Madjarov et al. [6] present an extensive study of different data-driven methods for constructing label hierarchies for multi-label classification by using the label co-occurrence matrix. More precisely, the hierarchies are constructed using four clustering algorithms, agglomerative clustering with single and complete linkage, balanced k -means and predictive clustering trees applied on the label co-occurrences (see Fig. 1, left table).

Next, Szymansky et al. [11] address the question whether data-driven approach using label co-occurrence graph is significantly better than a random choice

in label space division for multi-label classification as performed by RAKELd. Their results show that in almost all cases data-driven partitioning outperforms the baseline RAKELd in all evaluation measures, but Hamming loss.

In this study, we build upon the idea of decomposition of the output space and we present a different approach for data-driven structuring of label space in multi-label classification. Our approach constructs the label hierarchy by clustering the per label feature rankings. Namely, instead of using the original label space consisting of label co-occurrences (see Fig. 1, left table), we calculate a feature importance/ranking scores of the features for each label by using the GENIE3 method for feature importance calculation coupled with the random forest ensemble learning method [1,3] (see Fig. 1, right table).

The obtained structure is then used as the label hierarchy and the MLC task is addressed as hierarchical multi-label classification (HMLC) [5,15]. We thus evaluate whether considering the dependency in the label space can provide better predictive performance than addressing MLC as a flat problem. In other words, we investigate whether considering the MLC task as a hierarchical MLC task can yield better predictive performance. Our approach is illustrated through the example in Fig. 1. The table on the left hand-side shows the construction of the label hierarchy using the label co-occurrence (as performed in [6,11]), while the table on the right hand-side shows our proposed method for constructing the label hierarchy.

Input space				Output space of label co-occurrences						
	BH_LowPeakAmp	BH_LowPeakBPM	BH_HighPeakAmp	...	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
#1	0.036299	-58.962537	4.698083	...	1	0	0	0	0	0
#2	0.161218	-77.425609	3.09809	...	0	0	1	0	1	1
#3	0.115987	-61.893693	4.478436	...	1	1	1	1	0	0
#4	0.086016	-83.295694	3.786274	...	1	0	0	0	1	1
#5	0.063232	-76.108186	5.911183	...	0	1	0	1	1	1
#6	0.026461	-74.429498	3.046795	...	0	0	1	0	1	1
...

Structured label/output space						
	FRank λ_1	FRank λ_2	FRank λ_3	FRank λ_4	FRank λ_5	FRank λ_6
BH_LowPeakAmp	1.369	12.63	22.68	14.06	5.563	1.328
BH_LowPeakBPM	1.588	11.89	26.35	9.177	5.566	0.674
BH_HighPeakAmp	1.433	11.08	44	8.951	19.03	1.479
BH_HighPeakBPM	1.741	7.836	8.206	10.06	3.61	0.561
BH_HighLowRatio	2.169	7.267	6.914	9.166	12.16	0.017
BHSUM1	2.246	5.541	5.494	11.19	14.31	1.058
...

Fig. 1. Excerpt from the original *emotions* dataset showing the output space consists of label co-occurrences (*left table*) and the space consists of ranks of the features for each of the labels, separately (*right table*). The former is obtained with structuring the original label set using feature ranking.

We perform an experimental evaluation using 8 benchmark datasets from different domains: text, image, music and video classification, and gene function prediction. The predictive performance of the methods is assessed using 13 different evaluation measures used in the context of MLC (6 threshold dependent and 7 threshold independent).

The obtained results indicate that using the methods for creating the hierarchies using feature ranking can yield a better predictive performance as compared to the original flat MLC methods without the hierarchy. Moreover, using the hierarchy constructed by structuring of the output space using the feature rankings of the labels gives better predictive performance compared to using the hierarchy obtained using the label co-occurrences.

The remainder of this paper is organized as follows. Section 2 presents the background work, i.e., discussion on the tasks of multi-label classification and hierarchical multi-label classification methods. Then, in Sect. 3, we present the structuring of the output space using feature ranking. In Sect. 4, we show the experimental design. The results obtained from the experiments are presented and discussed in Sect. 5. Finally, Sect. 6 concludes this paper.

2 Background

In this section, we first define the task of multi-label classification and then the task of hierarchical multi-label classification. Multi-label learning considers learning from examples which are associated to more than one label coming from a predefined set of labels containing all possible labels. There are two types of multi-label learning tasks: multi-label classification and multi-label ranking. The main goal of multi-label classification is to create a predictive model that will output a set of relevant labels for a given, previously unseen example. Multi-label ranking, on the other hand, can be understood as learning a model that, for each unseen examples, associates a list of rankings (preferences) on the labels from a given set of possible labels and a bipartite partition of this set into relevant and irrelevant labels. An extensive bibliography of methods for multi-label learning can be found in [7, 14] and the references therein.

The task of multi-label learning can be defined as follows [5]. The input space \mathcal{X} consists of vectors of values of nominal or numeric data types i.e., $\forall x_i \in \mathcal{X}, x_i = (x_{i1}, x_{i2}, \dots, x_{iD})$, where D is a number of descriptive attributes. The output space \mathcal{Y} consists of a subset of a finite set of disjoint labels $\mathcal{L} = \{\lambda_1, \lambda_2, \dots, \lambda_Q\}$ ($Q > 1$ and $\mathcal{Y} \subseteq \mathcal{L}$). Given this, each example is a pair of a vector and a set from the input and output space, respectively. All of the examples then form the set of examples (i.e., the dataset) E . The goal is then to find a function $h : \mathcal{X} \rightarrow 2^{\mathcal{L}}$ such that from the input space assigns a set of labels to each example.

The main difference between multi-label classification and hierarchical multi-label classification (HMLC) is that in the latter the labels from the label space are organized into a hierarchy. A given example labeled with a given label it is also labeled with all its parent labels (known as the hierarchy constraint). Furthermore, an example can be labeled with multiple labels, simultaneously. That means a several paths can be followed from the root node in order to arrive at a given label.

Here, the output space \mathcal{Y} is defined with a label hierarchy (\mathcal{L}, \leq_h) , where \mathcal{L} is a set of labels and \leq_h is a partial order parent-child relationship structured as a tree ($\forall \lambda_1, \lambda_2 \in \mathcal{L} : \lambda_1 \leq_h \lambda_2$ if and only if λ_1 is a parent of λ_2) [5]. Each example from the set of examples E is a pair of a vector and a set from the input and output space respectively, where the set satisfies the hierarchy constraint, i.e., $E = \{(x_i, \mathcal{Y}_i) | x_i \in \mathcal{X}, \mathcal{Y} \subseteq \mathcal{L}, \lambda \in \mathcal{Y}_i \Rightarrow \forall \lambda' \leq_h \lambda : \lambda' \in \mathcal{Y}_i, 1 \leq i \leq N\}$, where N is a number of examples in E . Same conditions as in multi-label classification should be satisfied for the quality criterion q (high predictive performance and

low computational cost). In [9], an extensive bibliography is given, where the HMLC task is presented across different application domains.

3 Structuring of Label Spaces Using Feature Ranking

In this section, we explain our method for structuring the label space using feature ranking and we describe the different clustering algorithms used in this work. Our proposed method for label space structuring is outlined in procedure *StructuringLabelSpaceFR* in Table 1. First, we take the original training dataset D^{train} and using random forest method with GENIE3 feature importance, we create feature rankings for each label separately. We then construct a dataset D^{ranks} consisting of the feature rankings. Next, we obtain a hierarchy using one of the clustering algorithms described bellow. The hierarchy is then used to pre-process the datasets and obtain their hierarchical variants D_H^{train} and D_H^{test} . At the end, we learn the HMLC predictive models.

Table 1. The algorithm for structuring the label space using feature rankings per label.

```

procedure StructuringLabelSpaceFR( $D^{train}$ ,  $D^{test}$ ) returns performance
1: // create feature importance (.fimp) file with Random forest (GENIE3)
2: FimpPath = CreateFimp( $D^{train}$ );
3: // Create new arff with feature ranks from fimp file
4:  $D^{ranks}$  = CreateArffFromFimp(FimpPath);
5: hierarchy = Clustering( $D^{ranks}$ );
6: //transform multi-label dataset to hierarchical multi-label one
7:  $D_H^{train}$  = MLC2HMC( $D^{train}$ , hierarchy);
8:  $D_H^{test}$  = MLC2HMC( $D^{test}$ , hierarchy);
9: //solve transformed hierarchical multi-label problem by using approach for HMC
10: HMCModel = HMCMethod( $D_H^{train}$ );
11: //generate HMC predictions using CLUS platform
12: predictions = HMCModel( $D_H^{test}$ );
13: //Extract predictions only for the leaves from the HMC predictions
14: P = ExtractLeavesPredictionsFromHMCPredictions(predictions);
15: return Evaluate(P)

```

In our approach, described in a procedure *StructuringLabelSpaceFR* (Table 1), we can see that additional step, compare to the algorithm given by Madjarov et al. [6], is the function *CreateFimp* at line 4, which increases the theoretical complexity of the procedure. According to the dimensionality of the space which is going to be clustered using the function *Clustering* at line 5, one dimension in the space consists of label co-occurrences is the number of examples (instances) which means that in case of more complex datasets with large number of examples, the clustering procedure will take more of the time in order to create a hierarchy. From the other side, the procedure of creating the hierarchy

using feature rankings has a dimension which depends of the feature space cardinality. Typically, the feature space cardinality is much smaller than the number of examples. It means that clustering of the rankings will finish faster than clustering of the label co-occurrences for datasets with large number of examples but small number of features, which is a case in most of the benchmarks datasets available. Consequently, although we have additional function in our procedure of structuring of the output space, for more complex datasets with high number of examples and smaller number of features, the clustering procedure, i.e., the hierarchy creation will be completed in a reasonable time, thus compensating for obtaining the feature rankings.

We next describe the procedures for obtaining the feature rankings. Random forests as ensemble method for predictive modeling are originally proposed by Breiman [1]. The empirical analysis of their use as feature ranking methods has been studied by Verikas et al. [16]. The random forests are constructed by first performing bootstrap sampling on the data and then building a decision tree for each bootstrap sample. The decision trees are constructed by taking the best split at each level, from a randomly selected feature subset.

Huynh-Thu et al. [3] propose to use the reduction of the variance in the output space at each test node in the tree (the resulting algorithm is named GENIE3). Namely, the variables that reduce the variance of the output more are, consequently, more important than the ones that reduce the variance less. Hence, for each descriptive variable we measure the reduction of variance it produces when selected as splitting variable. If a variable is never selected as splitting variable then its importance will be 0.

The GENIE3 algorithm has been heavily evaluated for single-target regression tasks (e.g., for gene regulatory network reconstruction). The basic idea adopted for feature ranking is the same of that proposed in GENIE3, but we use random forest of predictive clustering trees (PCTs) for building the ensemble. The result is a feature ranking algorithm that works for different types of structure output prediction tasks (including MLC and HMLC).

Furthermore, we discuss the different clustering methods used to obtain the hierarchies of the labels. For achieving a good performance of the HMLC methods, it is critical to generate label hierarchies that more closely capture the relations among the labels. The only constraint when building the hierarchy is that we should take care about the leaves of the label hierarchies. They need to define the original MLC task. In particular, the labels from the original MLC problem represent the leaves of the label hierarchy, while the labels in internal nodes of the tree are so-called meta-labels. Meta-labels model the potential relations among the original labels.

For obtaining the hierarchies, we use four different clustering methods (two agglomerative and two divisive):

- agglomerative clustering with single linkage;
- agglomerative clustering with complete linkage;
- balanced k-means clustering (*divisive*) and
- predictive clustering trees (*divisive*).

Agglomerative clustering algorithms consider each example as separate cluster at the beginning and then iteratively merge pairs of clusters based on their distance metric (linkage). If we use the maximal distance of two examples from the clusters C_1 and C_2 , then this type of agglomerative clustering is using *complete* linkage, i.e., $\max\{dist(c_1, c_2) : c_1 \in C_1, c_2 \in C_2\}$. If we use the minimal distance between two clusters, then the agglomerative clustering approach is with *single* linkage i.e., $\min\{dist(c_1, c_2) : c_1 \in C_1, c_2 \in C_2\}$.

Balanced k-means is top-down approach for clustering. First, all labels from the label space \mathcal{L} are in one common cluster at the top node of the hierarchy. Then, the procedure consecutively divides (splits) this cluster into k disjoint sub-clusters ($k < |\mathcal{L}_n|$) using k-means clustering. The division also is concerned with the number of examples in each cluster: the algorithm outputs clusters with approximately equal size [13]. The procedure recursively is repeated on each sub-cluster (meta-label) until we have n different clusters consisting of one label from the label space \mathcal{L} . In other words, our label space \mathcal{L} is covered by leaves of the hierarchy obtained by the balanced k-means clustering approach.

We also use predictive clustering trees to construct the label hierarchies. More specifically, the setting from the predictive clustering framework used in this work is based on treating the target space as descriptive space, i.e., the target space is also a descriptive space. Descriptive/target variables are used to provide descriptions for the obtained clusters. Here, the focus is using predictive clustering framework on the task of clustering instead of predictive modelling [2, 4]. The obtained hierarchies using agglomerative clustering (single and complete linkage) and using predictive clustering trees for *emotions* dataset are shown in Fig. 2.

We next present the predictive clustering trees (PCTs) - the modelling framework we used throughout this work. PCTs are a generalization of decision trees towards the tasks of predicting structured outputs, including both MLC and HMLC. In order to apply PCTs to the task of HMLC, Vens et al. [15] define the variance and the prototype as follows. First, the set of labels for each example is represented as a vector of binary components. If the example belongs to the class c_i then the i 'th component of the vector is 1 and 0, otherwise. The variance of a set of examples E is thus defined as follows:

$$Var(E) = \frac{1}{|E|} \cdot \sum_{i=1}^{|E|} dist(\Gamma_i, \bar{\Gamma})^2 \quad (1)$$

where $\bar{\Gamma} = \frac{1}{|E|} \cdot \sum_{i=1}^{|E|} \Gamma_i$.

In other words, the variance $Var(E)$ in (1) represents the average squared distance between each example's class vector (Γ_i) and the mean class vector of the set ($\bar{\Gamma}$). When we talk about HMC, then the similarity at higher levels of the hierarchy are more important than the similarity at lower levels. This is reflected with the distance term used in (1), which is weighted Euclidean distance:

$$dist(\Gamma_1, \Gamma_2) = \sqrt{\sum_{s=1}^{|\Gamma|} \theta(c_s) \cdot (\Gamma_{1,s} - \Gamma_{2,s})^2}$$

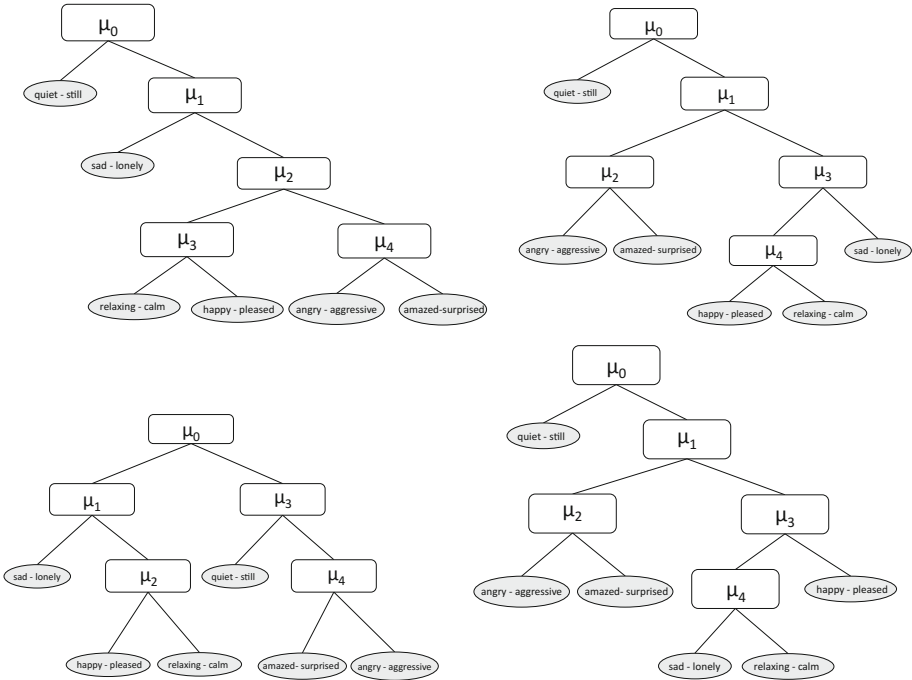


Fig. 2. Hierarchies obtained using agglomerative single (top-left), agglomerative complete (top-right), balanced K-means clustering (bottom - left) and PCTs (bottom - right) clustering methods for *emotions* dataset.

where $\Gamma_{i,s}$ is the s 'th component of the class vector Γ_i of the instance E_i , $|\Gamma|$ is the size of the class vector, and the class weights $\theta(c) = \theta_0 \cdot \text{avg}_j \{\theta(p_j(c))\}$, where $p_j(c)$ is j 'th parent of the class c and $0 < \theta_0 < 1$. The class weights $\theta(c)$ decrease with the depth of the class in the hierarchy thus making the differences in the lower parts of the hierarchy less influential to the overall score.

Random forests of PCTs for HMLC are considered in the same way as the random forest of PCTs for MLC. In the case of HMLC, the ensemble is a set of PCTs for HMLC. A new example is classified by taking a majority vote from the combined predictions of the member classifiers. The prediction of the random forest ensemble of PCTs for HMLC follows the hierarchy constraint (if the example is labeled with a given label then is automatically labeled with all its ancestor-labels).

4 Experimental Design

The aim of our study is to address the following questions:

- (i) Whether feature ranking on the label (output) space in the MLC task can be used to construct good label hierarchies?

- (ii) Which clustering method yields better hierarchy?
- (iii) How this scales from single model to ensemble of models?
- (iv) Can we achieve better predictive models with using a hierarchies obtained by structuring the feature ranking or co-occurrences space?

In order to answer the above questions, we use eight multi-label classification benchmark problems from different domains. We have 3 datasets from text classification, 4 datasets from multimedia, includes movie clips and genres classification and 1 dataset from biology. All datasets are predefined by other researchers (typically the data owners) and divided into train and test subsets. The basic information and statistics about these datasets are given in Table 2.

Table 2. Statistics of used benchmark tasks in terms of application domain (*domain*), number of training examples (*#tr.e*), testing examples (*#t.e*), number of descriptors (*D*), total number of labels (*L*) and number of labels per example.

Dataset	Domain	<i>#tr.e</i>	<i>#t.e</i>	<i>D</i>	<i>L</i>	<i>l_c</i>
emotions	multimedia	391	202	72	6	1.87
scene	multimedia	1211	1159	294	6	1.07
yeast	biology	1500	917	103	14	4.24
tmc2007	text	21519	7077	500	22	2.16
medical	text	645	333	1449	45	1.25
enron	text	1123	579	1001	53	3.38
mediamill	multimedia	30993	12914	120	101	4.38
corel5k	multimedia	4500	500	499	374	3.52

In our experiments, we use 13 different evaluation measures, as presented in [7, 14]. These are divided into two groups: 6 threshold dependent/example based measures (*hamming loss*, *accuracy*, *precision*, *recall*, *F₁ score*) and 7 threshold independent measures out of which three ranking-based (*one-error*, *coverage* and *ranking-loss*) and four areas under ROC and PRC curves (*AUROC*, *AUPRC*, *wAUPRC* and *pooledAUPRC*). The threshold independent measures are typically used in HMLC and they do not require a (pre)selection of thresholds and calculating a prediction [15]. All of the above measures offer different viewpoints on the results from the experimental evaluation.

Hamming loss is an example-based evaluation measure that evaluate how many times a pair of example and its label are misclassified. *One-error* is a ranking-based evaluation measure that evaluates how many times the top-ranked label does not exist in a set of relevant labels of the example. *Coverage* evaluates how far, on average, we need to go down the list of label ranks in order to cover all relevant labels of given example. *Ranking loss* evaluates the average fraction of the label pairs that are reversely ordered for the given example. Precision and recall are very important measures defined for binary classification tasks with

classes of positive and negative examples. *Precision* is a proportion of positive prediction that are correct, and *recall* is the proportion of positive examples that correctly predicted as positive. F_1 score is the harmonic mean between precision and recall. *Accuracy* for each instance is defined as the proportion of correctly predicted labels over total number of labels for that instance. Overall accuracy is the average across all instances. A precision-recall curve (PR curve) is a curve that represent the precision as a function of its recall. *AUPRC* (area under the PR curve) is the area between the PR curve and the recall axis. *wAUPRC* evaluates the weighted average of the areas under the individual (per class) PR-curves. If choosing some threshold, we transform the multi-label problem into binary problems with considering binary classifier as a couple (instance, class) and predicting whether that instance belongs to that class, we can obtain PR curves that differ depend of the varying the threshold. The area under the average PR curve (from all different threshold curves) is called *pooledAUPRC*. From the other side, if we consider the space of true positive rates (sensitivity) versus false positive rates (fall-out) then the curve considers the sensitivity as a function of the fall-out is called ROC-curve. The are under this ROC-curve is the evaluation measure called *AUROC*.

The majority of our experiments are performed using the CLUS software package (<https://sourceforge.net/projects/clus/>), which implements the predictive clustering framework, including PCTs, random forests of PCTs and feature ranking [5,10]. A hierarchical tree defined by the used clustering methods in HMLC setting are defined as tree shaped hierarchies. We use the same values for k in balanced k-means clustering algorithm, as suggested in [7].

For obtaining a hierarchy using the agglomerative clustering method we use the R software package (function *agnes()* from the *cluster* package. For more info, see <https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/agnes.html>). We use the MATLAB software package to create hierarchies with balanced k-means clustering which is based on Hungarian (Munkres') assignment algorithm to assign the examples to the clusters [8]. We use Euclidean distance metric in all our algorithms that require distance. Moreover, for random forest for feature ranking we use GENIE3 as a feature importance method based on variable selection with ensembles of PCTs [3].

In order to make a comparative analysis with the results obtained by the study by Madjarov et al. [6], we repeated their experiments on the same experimental setting with the experiments we perform for feature ranking.

5 Results

In this section, we present the obtained results from the experiments we performed using our novel proposed method for structuring the output space. In our study, as an output space, we consider the space consisting of label co-occurrences (as presented by Madjarov et al. [6]) and the space consisting of feature ranks for each label, respectively. We compare the following methods for hierarchy construction:

- flat MLC problem without considering a hierarchy in the label space (*FlatMLC*);
- agglomerative clustering with single linkage (*AggSingle*);
- agglomerative clustering with complete linkage (*AggComplete*);
- balanced k-means clustering (*BKmeans*);
- clustering using predictive clustering trees (*ClusPCTs*).

Since we have two different models (single PCTs model and random forest of PCTs) and two different structured output spaces, we show separately the results for single PCTs (Fig. 3) and random forest of PCTs (Fig. 4). In order to distinguish between using either single tree or random forest of PCTs and different methods of structuring the output space (label co-occurrences and feature rankings), we use prefixes (*PCT-* and *RF-*) and suffixes (*-CO* and *-FR*) before and after the hierarchy construction method name, respectively. For example, *RF-AggComplete-CO* refers to the agglomerative clustering method with complete linkage of the output space of label co-occurrences using random forest of PCTs for model creation. Then, *PCT-ClusPCTs-FR* refers to the clustering method with PCTs of the output space consists of feature rankings per label using single PCTs for model creation, etc.

Observing the results obtained using single PCTs (Fig. 3), we can note that there is no clear winner across all evaluation measures and datasets. In the case of threshold independent measures, such as *AUPRC*, *AUROC*, *wAUPRC* and *pooledAUPRC*, we can see that hierarchies created using clustering of the output space consisting of feature rankings perform the best for enron, emotions, mediamill and yeast datasets. Considering the scene and corel5k datasets, we can observe that they perform the best according to *AUROC*, *AUPRC* and *pooledAUPRC*, but not for *wAUPRC*. *PCT-BKmeans-FR* outperforms the other algorithms for hierarchy creation in the emotions dataset according to the most of the evaluation measures but not according to one-error. Moreover, the hierarchies created clustering the feature rankings outperform the other algorithms considering the ML performance measures (*ML F1 measure*, *ML accuracy*, *ML precision* and *ML recall*) in 5 out of the 8 datasets.

Generally, structuring the output space consisting of feature rankings for each label yields better predictive performance compared to the structuring the output space consisting of label co-occurrences considering most of the evaluation measures in almost all datasets. For the corel5k dataset only, we can see that both have similar performance. If we consider medical and tmc2007 datasets, we can see that structuring the output space does not improve the performance as compared to the flat MLC task, where there is no hierarchy considered. All in all, we can conclude that using the hierarchies, the predictive performance can be improved.

The results obtained when random forests are used as predictive models are given in Fig. 4. These results present a different situation as compared to the results obtained when single PCTs are used as predictive models. First of all, the predictive performance is improved as compared to the single PCTs for large majority of the cases. Most notably, the performance for the threshold

PCTs	Hamming loss	Average precision	Coverage	ML Accuracy	ML F1 measure	ML Precision	ML Recall	One Error	Ranking Loss	AUROC	AUPRC	wAUPRC	pooler AUPRC
ENRON													
PCT-FlatMLC	0.071	0.538	40.513	0.360	0.467	0.489	0.502	0.444	0.151	0.130	0.585	0.353	0.416
PCT-AggSingle-FR	0.071	0.595	39.630	0.380	0.485	0.505	0.527	0.383	0.104	0.142	0.598	0.367	0.428
PCT-AggComplete-FR	0.072	0.565	39.703	0.371	0.478	0.486	0.530	0.382	0.192	0.148	0.601	0.370	0.433
PCT-BKmeans-FR	0.072	0.466	39.665	0.374	0.480	0.489	0.527	0.501	0.341	0.142	0.593	0.358	0.419
PCT-ClusterPCTs-FR	0.072	0.554	39.472	0.354	0.459	0.471	0.499	0.382	0.194	0.142	0.590	0.354	0.418
PCT-AggSingle-CO	0.067	0.482	36.858	0.374	0.475	0.488	0.520	0.458	0.356	0.137	0.591	0.362	0.421
PCT-AggComplete-CO	0.066	0.471	37.104	0.364	0.463	0.485	0.504	0.453	0.350	0.128	0.580	0.359	0.419
PCT-BKmeans-CO	0.068	0.541	37.879	0.364	0.472	0.493	0.522	0.393	0.222	0.131	0.586	0.357	0.413
PCT-ClusterPCTs-CO	0.072	0.588	40.473	0.374	0.476	0.487	0.517	0.396	0.108	0.142	0.594	0.366	0.424
EMOTIONS													
PCT-FlatMLC	0.292	0.669	4.431	0.460	0.541	0.550	0.582	0.450	0.335	0.516	0.680	0.509	0.524
PCT-AggSingle-FR	0.304	0.666	4.569	0.421	0.502	0.514	0.549	0.490	0.317	0.487	0.661	0.480	0.503
PCT-AggComplete-FR	0.296	0.672	4.574	0.442	0.528	0.551	0.559	0.470	0.314	0.507	0.679	0.508	0.517
PCT-BKmeans-FR	0.266	0.717	4.173	0.507	0.589	0.597	0.634	0.401	0.265	0.552	0.714	0.558	0.563
PCT-ClusterPCTs-FR	0.292	0.702	4.569	0.438	0.529	0.554	0.564	0.388	0.291	0.505	0.670	0.509	0.517
PCT-AggSingle-CO	0.307	0.670	4.460	0.439	0.525	0.528	0.576	0.450	0.345	0.496	0.664	0.495	0.510
PCT-AggComplete-CO	0.307	0.670	4.460	0.439	0.525	0.528	0.576	0.450	0.345	0.496	0.664	0.495	0.510
PCT-BKmeans-CO	0.312	0.640	4.698	0.414	0.507	0.541	0.535	0.485	0.357	0.496	0.653	0.491	0.505
PCT-ClusterPCTs-CO	0.297	0.681	4.639	0.440	0.516	0.535	0.535	0.446	0.323	0.489	0.664	0.491	0.505
MEDICAL													
PCT-FlatMLC	0.014	0.795	11.447	0.724	0.766	0.759	0.809	0.204	0.104	0.321	0.686	0.672	0.702
PCT-AggSingle-FR	0.015	0.794	12.874	0.706	0.741	0.742	0.771	0.216	0.082	0.320	0.685	0.646	0.682
PCT-AggComplete-FR	0.015	0.785	12.207	0.721	0.759	0.758	0.791	0.222	0.115	0.325	0.690	0.665	0.692
PCT-BKmeans-FR	0.015	0.771	12.616	0.710	0.750	0.751	0.786	0.219	0.125	0.320	0.689	0.648	0.687
PCT-ClusterPCTs-FR	0.015	0.787	11.832	0.727	0.767	0.774	0.803	0.231	0.087	0.314	0.696	0.670	0.699
PCT-AggSingle-CO	0.016	0.761	12.258	0.694	0.733	0.726	0.777	0.264	0.133	0.315	0.684	0.645	0.687
PCT-AggComplete-CO	0.016	0.763	12.640	0.694	0.734	0.733	0.773	0.240	0.141	0.294	0.662	0.638	0.676
PCT-BKmeans-CO	0.015	0.795	12.003	0.716	0.757	0.753	0.797	0.198	0.078	0.340	0.695	0.652	0.691
PCT-ClusterPCTs-CO	0.016	0.795	12.003	0.707	0.747	0.751	0.783	0.228	0.068	0.298	0.678	0.658	0.686
MEDIAMILL													
PCT-FlatMLC	0.052	0.472	77.282	0.356	0.476	0.491	0.551	0.445	0.247	0.089	0.571	0.339	0.440
PCT-AggSingle-FR	0.052	0.584	76.868	0.353	0.474	0.495	0.549	0.318	0.105	0.087	0.570	0.350	0.439
PCT-AggComplete-FR	0.052	0.610	76.795	0.358	0.478	0.498	0.553	0.313	0.083	0.089	0.570	0.353	0.443
PCT-BKmeans-FR	0.053	0.509	76.514	0.357	0.477	0.493	0.554	0.394	0.118	0.093	0.574	0.347	0.441
PCT-ClusterPCTs-FR	0.052	0.604	76.004	0.360	0.479	0.498	0.552	0.351	0.074	0.088	0.574	0.352	0.443
PCT-AggSingle-CO	0.053	?	73.362	0.341	0.452	0.478	0.516	0.440	0.291	0.087	0.562	0.345	0.429
PCT-AggComplete-CO	0.055	?	72.275	0.339	0.450	0.474	0.513	0.516	0.321	0.081	0.564	0.337	0.428
PCT-BKmeans-CO	0.054	?	70.465	0.349	0.463	0.479	0.537	0.471	0.273	0.090	0.571	0.339	0.434
PCT-ClusterPCTs-CO	0.051	?	78.356	0.343	0.455	0.480	0.516	0.267	0.156	0.088	0.569	0.339	0.428
SCENE													
PCT-FlatMLC	0.263	0.636	4.537	0.271	0.288	0.289	0.302	0.686	0.183	0.193	0.530	0.255	0.907
PCT-AggSingle-FR	0.251	0.491	4.215	0.311	0.333	0.332	0.360	0.669	0.475	0.183	0.479	0.282	0.903
PCT-AggComplete-FR	0.247	0.658	4.595	0.304	0.333	0.351	0.347	0.628	0.166	0.191	0.494	0.265	0.907
PCT-BKmeans-FR	0.237	0.688	4.157	0.342	0.371	0.372	0.397	0.587	0.151	0.196	0.546	0.291	0.906
PCT-ClusterPCTs-FR	0.247	0.470	4.595	0.304	0.333	0.351	0.347	0.661	0.525	0.191	0.494	0.265	0.907
PCT-AggSingle-CO	0.234	0.557	4.256	0.349	0.376	0.373	0.405	0.579	0.361	0.189	0.516	0.299	0.906
PCT-AggComplete-CO	0.234	0.557	4.256	0.349	0.376	0.373	0.405	0.579	0.361	0.189	0.516	0.299	0.906
PCT-BKmeans-CO	0.229	0.509	4.059	0.355	0.387	0.386	0.421	0.612	0.523	0.186	0.502	0.216	0.904
PCT-ClusterPCTs-CO	0.260	0.658	4.438	0.280	0.309	0.303	0.343	0.636	0.164	0.186	0.517	0.260	0.904
TMC2007													
PCT-FlatMLC	0.098	0.957	2.690	0.807	0.866	0.842	0.942	0.044	0.007	0.907	0.994	0.962	0.935
PCT-AggSingle-FR	0.030	0.948	2.712	0.797	0.859	0.835	0.936	0.052	0.009	0.905	0.993	0.955	0.950
PCT-AggComplete-FR	0.030	0.949	2.705	0.802	0.862	0.836	0.940	0.052	0.009	0.903	0.993	0.955	0.950
PCT-BKmeans-FR	0.029	0.950	2.648	0.807	0.867	0.842	0.943	0.053	0.008	0.928	0.993	0.959	0.955
PCT-ClusterPCTs-FR	0.030	0.950	2.684	0.801	0.862	0.837	0.940	0.048	0.009	0.903	0.993	0.956	0.949
PCT-AggSingle-CO	0.031	0.943	2.739	0.794	0.855	0.837	0.928	0.057	0.010	0.861	0.992	0.953	0.939
PCT-AggComplete-CO	0.030	0.946	2.711	0.797	0.859	0.835	0.937	0.056	0.009	0.870	0.992	0.955	0.942
PCT-BKmeans-CO	0.029	0.954	2.640	0.807	0.866	0.840	0.943	0.049	0.008	0.903	0.993	0.960	0.953
PCT-ClusterPCTs-CO	0.030	0.947	2.719	0.800	0.860	0.841	0.932	0.051	0.009	0.884	0.992	0.955	0.945
YEAST													
PCT-FlatMLC	0.295	0.630	11.124	0.406	0.514	0.516	0.572	0.430	0.299	0.354	0.558	0.483	0.510
PCT-AggSingle-FR	0.290	0.590	11.122	0.429	0.541	0.545	0.600	0.510	0.367	0.365	0.574	0.500	0.528
PCT-AggComplete-FR	0.289	0.608	11.109	0.417	0.526	0.529	0.584	0.507	0.320	0.368	0.578	0.504	0.527
PCT-BKmeans-FR	0.291	0.645	11.372	0.412	0.523	0.533	0.570	0.430	0.261	0.357	0.565	0.488	0.521
PCT-ClusterPCTs-FR	0.292	0.645	11.298	0.415	0.518	0.531	0.561	0.455	0.257	0.358	0.560	0.491	0.525
PCT-AggSingle-CO	0.298	0.648	11.262	0.408	0.516	0.521	0.573	0.353	0.317	0.356	0.556	0.491	0.517
PCT-AggComplete-CO	0.290	0.676	11.144	0.419	0.528	0.530	0.590	0.328	0.263	0.359	0.570	0.502	0.520
PCT-BKmeans-CO	0.288	0.670	11.352	0.412	0.519	0.530	0.566	0.334	0.275	0.363	0.568	0.498	0.523
PCT-ClusterPCTs-CO	0.296	0.668	11.241	0.412	0.520	0.526	0.575	0.400	0.246	0.355	0.558	0.497	0.518
COREL5K													
PCT-FlatMLC	0.015	0.144	352.716	0.091	0.130	0.175	0.125	0.774	0.419	0.027	0.516	0.058	0.114
PCT-AggSingle-FR	0.014	0.187	357.244	0.083	0.124	0.195	0.118	0.752	0.223	0.022	0.514	0.045	0.094
PCT-AggComplete-FR	0.016	0.184	354.216	0.083	0.121	0.142	0.125	0.734	0.409	0.021	0.513	0.055	0.106
PCT-BKmeans-FR	0.016	0.137	360.022	0.092	0.136	0.169	0.142	0.752	0.606	0.031	0.521	0.060	0.115
PCT-ClusterPCTs-FR	0.016	0.217	350.488	0.093	0.134	0.144	0.150	0.716	0.215	0.032	0.523	0.064	0.123
PCT-AggSingle-CO	0.013	0.096	368.088	0.065	0.097	0.169	0.085	0.778	0.712	0.013	0.501	0.037	0.083
PCT-AggComplete-CO	0.013	0.110	367.356	0.073	0.108	0.186	0.095	0.776	0.645	0.020	0.504	0.042	0.092
PCT-BKmeans-CO	0.015	0.181	351.246	0.101	0.147	0.168	0.156	0.700	0.294	0.029	0.518		

Random Forest	Hamming loss	Average precision	Coverage	ML Accuracy	ML F1 measure	ML Precision	ML Recall	One Error	Ranking Loss	AUROC	AUPRC	wAUPRC	pooled AUPRC
ENRON													
RF-FlatMLC	0.047	0.698	13.187	0.402	0.509	0.714	0.435	0.200	0.078	0.241	0.709	0.620	0.577
RF-AggSingle+FR	0.047	0.696	13.028	0.396	0.500	0.706	0.425	0.206	0.077	0.235	0.724	0.615	0.574
RF-AggComplete+FR	0.047	0.695	13.347	0.396	0.499	0.703	0.425	0.206	0.078	0.239	0.724	0.618	0.575
RF-BKmeans+FR	0.046	0.697	12.865	0.404	0.509	0.708	0.434	0.211	0.076	0.242	0.745	0.622	0.582
RF-ClusterPCTs+FR	0.046	0.696	13.180	0.402	0.506	0.704	0.431	0.199	0.076	0.244	0.737	0.620	0.582
RF-AggSingle+CO	0.042	0.686	11.784	0.405	0.507	0.726	0.424	0.193	0.079	0.213	0.728	0.598	0.553
RF-AggComplete+CO	0.042	0.692	11.717	0.410	0.511	0.719	0.430	0.202	0.079	0.215	0.730	0.603	0.559
RF-BKmeans+CO	0.043	0.688	12.223	0.399	0.503	0.728	0.420	0.200	0.078	0.225	0.719	0.600	0.554
RF-ClusterPCTs+CO	0.047	0.692	13.100	0.400	0.504	0.706	0.429	0.199	0.078	0.236	0.742	0.616	0.572
EMOTIONS													
RF-FlatMLC	0.191	0.813	2.812	0.530	0.605	0.674	0.600	0.267	0.152	0.755	0.851	0.754	0.755
RF-AggSingle+FR	0.201	0.815	2.837	0.500	0.569	0.629	0.567	0.282	0.155	0.749	0.852	0.756	0.753
RF-AggComplete+FR	0.196	0.810	2.817	0.502	0.574	0.643	0.564	0.262	0.151	0.766	0.859	0.762	0.769
RF-BKmeans+FR	0.199	0.810	2.817	0.494	0.563	0.626	0.553	0.277	0.153	0.770	0.863	0.767	0.772
RF-ClusterPCTs+FR	0.205	0.814	2.827	0.487	0.559	0.623	0.550	0.282	0.154	0.754	0.856	0.756	0.754
RF-AggSingle+CO	0.199	0.817	2.812	0.504	0.578	0.645	0.572	0.287	0.150	0.755	0.858	0.758	0.757
RF-AggComplete+CO	0.199	0.817	2.812	0.504	0.578	0.645	0.572	0.287	0.150	0.755	0.858	0.758	0.757
RF-BKmeans+CO	0.193	0.815	2.871	0.510	0.580	0.649	0.569	0.297	0.160	0.759	0.854	0.765	0.762
RF-ClusterPCTs+CO	0.191	0.820	2.787	0.512	0.582	0.648	0.575	0.267	0.148	0.764	0.860	0.766	0.766
MEDICAL													
RF-FlatMLC	0.018	0.858	2.571	0.415	0.431	0.462	0.418	0.396	0.023	0.432	0.824	0.787	0.818
RF-AggSingle+FR	0.019	0.856	2.700	0.356	0.371	0.402	0.359	0.459	0.024	0.439	0.812	0.764	0.803
RF-AggComplete+FR	0.018	0.865	2.589	0.417	0.434	0.470	0.418	0.402	0.022	0.458	0.820	0.790	0.828
RF-BKmeans+FR	0.018	0.865	2.589	0.430	0.447	0.479	0.435	0.393	0.023	0.467	0.823	0.795	0.831
RF-ClusterPCTs+FR	0.019	0.849	2.769	0.388	0.405	0.441	0.391	0.411	0.026	0.422	0.805	0.777	0.818
RF-AggSingle+CO	0.019	0.853	2.841	0.366	0.382	0.416	0.367	0.447	0.027	0.437	0.804	0.771	0.813
RF-AggComplete+CO	0.019	0.852	2.727	0.369	0.386	0.420	0.372	0.438	0.025	0.432	0.817	0.764	0.808
RF-BKmeans+CO	0.018	0.853	2.613	0.421	0.440	0.477	0.424	0.372	0.023	0.455	0.822	0.786	0.821
RF-ClusterPCTs+CO	0.019	0.857	2.586	0.376	0.397	0.438	0.379	0.423	0.023	0.441	0.813	0.778	0.818
MEDIAMILL													
RF-FlatMLC	0.030	0.735	20.676	0.455	0.573	0.798	0.495	0.124	0.047	0.254	0.762	0.671	0.618
RF-AggSingle+FR	0.030	0.733	20.781	0.451	0.570	0.803	0.489	0.124	0.047	0.249	0.765	0.669	0.617
RF-AggComplete+FR	0.030	0.733	20.727	0.451	0.569	0.802	0.487	0.127	0.047	0.254	0.765	0.668	0.616
RF-BKmeans+FR	0.030	0.735	20.546	0.453	0.571	0.800	0.491	0.124	0.046	0.252	0.773	0.671	0.617
RF-ClusterPCTs+FR	0.030	0.734	20.806	0.451	0.569	0.801	0.488	0.126	0.047	0.248	0.765	0.668	0.616
RF-AggSingle+CO	0.031	?	19.722	0.438	0.549	0.777	0.470	0.150	0.047	0.242	0.756	0.657	0.607
RF-AggComplete+CO	0.032	?	19.117	0.440	0.551	0.777	0.471	0.150	0.047	0.249	0.761	0.659	0.610
RF-BKmeans+CO	0.032	?	18.830	0.440	0.551	0.772	0.474	0.153	0.046	0.249	0.768	0.659	0.609
RF-ClusterPCTs+CO	0.030	?	20.681	0.434	0.546	0.775	0.465	0.152	0.045	0.248	0.763	0.656	0.607
SCENE													
RF-FlatMLC	0.169	0.631	2.405	0.202	0.204	0.207	0.202	0.339	0.247	0.193	0.515	0.457	0.908
RF-AggSingle+FR	0.174	0.608	2.496	0.174	0.174	0.174	0.174	0.347	0.272	0.186	0.495	0.440	0.904
RF-AggComplete+FR	0.174	0.624	2.314	0.174	0.174	0.174	0.174	0.331	0.234	0.189	0.502	0.434	0.905
RF-BKmeans+FR	0.172	0.646	2.298	0.198	0.198	0.198	0.198	0.364	0.231	0.189	0.519	0.456	0.905
RF-ClusterPCTs+FR	0.174	0.624	2.314	0.174	0.174	0.174	0.174	0.331	0.234	0.189	0.502	0.434	0.905
RF-AggSingle+CO	0.174	0.590	2.496	0.140	0.140	0.140	0.140	0.298	0.274	0.187	0.507	0.415	0.904
RF-AggComplete+CO	0.174	0.590	2.496	0.140	0.140	0.140	0.140	0.298	0.274	0.187	0.507	0.415	0.904
RF-BKmeans+CO	0.172	0.589	2.595	0.182	0.182	0.182	0.182	0.306	0.292	0.191	0.512	0.434	0.905
RF-ClusterPCTs+CO	0.169	0.614	2.545	0.182	0.182	0.182	0.182	0.339	0.279	0.190	0.513	0.434	0.905
TMC2007													
RF-FlatMLC	0.025	0.976	2.201	0.796	0.848	0.933	0.813	0.039	0.003	0.999	0.999	0.978	0.992
RF-AggSingle+FR	0.025	0.976	2.305	0.796	0.848	0.938	0.812	0.039	0.003	0.993	0.999	0.974	0.992
RF-AggComplete+FR	0.025	0.976	2.305	0.794	0.849	0.933	0.815	0.038	0.003	0.993	0.999	0.974	0.992
RF-BKmeans+FR	0.025	0.977	2.303	0.797	0.849	0.933	0.815	0.039	0.004	0.993	0.999	0.975	0.991
RF-ClusterPCTs+FR	0.026	0.976	2.309	0.789	0.842	0.931	0.805	0.042	0.003	0.992	0.999	0.973	0.992
RF-AggSingle+CO	0.027	0.976	2.309	0.776	0.831	0.928	0.790	0.044	0.004	0.993	0.998	0.973	0.994
RF-AggComplete+CO	0.031	0.947	2.749	0.795	0.877	0.834	0.931	0.052	0.009	0.872	0.992	0.954	0.941
RF-BKmeans+CO	0.025	0.976	2.305	0.791	0.844	0.931	0.808	0.040	0.003	0.993	0.999	0.975	0.992
RF-ClusterPCTs+CO	0.026	0.976	2.308	0.788	0.842	0.933	0.805	0.041	0.003	0.993	0.999	0.974	0.992
YEAST													
RF-FlatMLC	0.197	0.759	7.126	0.482	0.587	0.741	0.530	0.241	0.166	0.508	0.710	0.722	0.676
RF-AggSingle+FR	0.199	0.755	7.308	0.471	0.578	0.743	0.514	0.241	0.170	0.501	0.699	0.717	0.669
RF-AggComplete+FR	0.200	0.753	7.269	0.469	0.576	0.740	0.513	0.246	0.172	0.500	0.682	0.713	0.665
RF-BKmeans+FR	0.199	0.755	7.215	0.473	0.580	0.737	0.521	0.248	0.167	0.505	0.704	0.716	0.669
RF-ClusterPCTs+FR	0.198	0.755	7.252	0.477	0.583	0.739	0.524	0.244	0.169	0.504	0.692	0.714	0.669
RF-AggSingle+CO	0.198	0.757	7.201	0.479	0.586	0.742	0.530	0.242	0.168	0.506	0.699	0.719	0.673
RF-AggComplete+CO	0.196	0.759	7.218	0.484	0.591	0.742	0.535	0.240	0.167	0.511	0.707	0.717	0.674
RF-BKmeans+CO	0.196	0.759	7.215	0.483	0.588	0.740	0.529	0.246	0.166	0.508	0.698	0.719	0.674
RF-ClusterPCTs+CO	0.199	0.758	7.217	0.474	0.581	0.738	0.522	0.241	0.168	0.503	0.695	0.716	0.671
CORELSK													
RF-FlatMLC	0.009	0.317	103.856	0.016	0.025	0.056	0.016	0.298	0.107	0.068	0.656	0.200	0.230
RF-AggSingle+FR	0.009	0.298	105.210	0.020	0.030	0.069	0.020	0.236	0.109	0.066	0.658	0.185	0.229
RF-AggComplete+FR	0.009	0.319	101.606	0.015	0.023	0.052	0.015	0.306	0.107	0.068	0.660	0.208	0.236
RF-BKmeans+FR	0.009	0.327	102.092	0.012	0.018	0.042	0.012	0.320	0.106	0.067	0.665	0.219	0.236
RF-ClusterPCTs+FR	0.009	0.313	107.224	0.017	0.026	0.058	0.017	0.286	0.110	0.070	0.654	0.201	0.234
RF-AggSingle+CO	0.009	0.266	121.804	0.020	0.031	0.072	0.020	0.206	0.127	0.061	0.636	0.155	0.215
RF-AggComplete+CO	0.009	0.269	120.950	0.021	0.032	0.074	0.021	0.228	0.126	0.064	0.636	0.155	0.218
RF-BKmeans+CO	0.009	0.343	97.858	0.014	0.022	0.047	0.014	0.364	0.101	0.075	0.674	0.227	0.245
RF-ClusterPCTs+CO	0.009	0.301	106.638	0.017	0.027	0.062	0.017	0.264	0.109	0.066	0.654	0.186	0.224

Fig. 4. Results with the 13 performance measures for *Random Forest* from experiments performed on 8 different datasets. The best results obtained per measure per dataset are highlighted.

independent measures (*AUPRC*, *AUROC*, *wAUPRC* and *pooledAUPRC*) for the mediamill and tmc2007 datasets are improved for almost twice, which is consistent to the notion from the literature that ensembles of PCTs improve the performance over single predictive models. Hierarchies created with clustering of the space consisting of feature rankings outperform both hierarchies obtained using label co-occurrences and flat MLC for the threshold independent measures on the medical, enron and emotions datasets. *RF-BKmeans-FR* performs the best for medical dataset in seven evaluation measures. Considering the hierarchies obtained with clustering the space of label co-occurrences, we can note that they outperform the other methods for the corel5k dataset. Using hierarchies (i.e., label dependences) rather than flat multi-label task improves the predictive performance generally for most of the evaluation measures, but not for (*ML F1 measure*, *ML accuracy*, *ML precision* and *ML recall*) in the emotions and scene datasets.

Finally, in our study we also considered training errors i.e., the errors made in the learning phase. There, in a large majority of the cases, the original *FlatMLC* method performed the best. This means that other methods we use for constructing the hierarchies do not overfit as the original one. This is another advantage of methods for construction the hierarchies identified from the obtained results.

6 Conclusions and Further Work

In this work, we have presented an approach for hierarchy construction and structuring the output (label) space by using feature ranking. More specifically, we cluster the feature rankings to obtain a hierarchical representation of the potential relations existing among the different labels. We then address the task of MLC as a task of HMLC. Moreover, we compare our approach with the approach of clustering the space consisting of label co-occurrences [6].

We investigated four clustering methods for hierarchy creation, agglomerative clustering with single and complete linkage, balanced k-means and clustering using predictive clustering trees (PCTs). The resulting problem was then approached as a HMLC problem using PCTs and random forests of PCTs for HMLC. We used eight benchmark datasets to evaluate the performance.

The results reveal that the best methods for hierarchy construction are agglomerative clustering methods and balanced k-means. Compared to the original MLC method where there is no hierarchy this improves the performance in most of the datasets. In four datasets, the hierarchies obtained by clustering the label space consisting of feature rankings improve the predictive performance compared to the hierarchies obtained by clustering the space consisting of label co-occurrences. Similar conclusions, but to a lesser extent, can be made for the random forests of PCTs for HMLC - in many of the cases (datasets and evaluation measures) the predictive models exploiting the hierarchy of labels yield better predictive performance. Finally, by considering the training error performance, we find that original MLC models overfit more than the HMLC models.

For further work, we plan to make more extensive evaluation on more datasets with diverse properties and to try more different feature ranking methods. Furthermore, we assume that potential improvement of the performance can be achieved with cutting the hierarchies based on some conditions such as density, distribution or distance between nodes. Moreover, we plan to include a comparison to network approaches given by Szymanski et al. [11]. Finally, we plan to extend this approach to other tasks, such as multi-target regression.

Acknowledgment. We would like to acknowledge the support of the European Commission through the project MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data (Grant number ICT-2013-612944), the project LAND-MARK - Land management, assessment, research, knowledge base (H2020 Grant number 635201) and Teagasc Walsh Fellowship Programme.

References

1. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
2. Dimitrovski, I., Kocev, D., Loskovska, S., Džeroski, S.: Fast and scalable image retrieval using predictive clustering trees. In: International Conference on Discovery Science, pp. 33–48 (2013)
3. Huynh-Thu, V.A., Irrthum, Wehenkel, L., Geurts, P.: Inferring regulatory networks from expression data using tree-based methods. *PLoS One* **5**(9) (2010)
4. Kocev, D.: Ensembles for predicting structured outputs. Ph.D. thesis, IPS Jožef Stefan, Ljubljana, Slovenia (2011)
5. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. *Pattern Recogn.* **46**(3), 817–833 (2013)
6. Madjarov, G., Dimitrovski, I., Gjorgjevikj, D., Džeroski, S.: Evaluation of different data-derived label hierarchies in multi-label classification. In: International Workshop on New Frontiers in Mining Complex Patterns, pp. 19–37 (2014)
7. Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S.: An extensive experimental comparison of methods for multi-label learning. *Pattern Recogn.* **45**(9), 3084–3104 (2012)
8. Malinen, M.I., Fränti, P.: Balanced K -means for clustering. In: Fränti, P., Brown, G., Loog, M., Escolano, F., Pelillo, M. (eds.) *S+SSPR 2014*. LNCS, vol. 8621, pp. 32–41. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44415-3_4
9. Silla, C.N., Freitas, A.: A survey of hierarchical classification across different application domains. *Data Min. Knowl. Disc.* **22**, 31–72 (2011)
10. Struyf, J., Džeroski, S.: Constraint based induction of multi-objective regression trees. In: Bonchi, F., Boulicaut, J.-F. (eds.) *KDID 2005*. LNCS, vol. 3933, pp. 222–233. Springer, Heidelberg (2006). https://doi.org/10.1007/11733492_13
11. Szymanski, P., Kajdanowicz, T., Kersting, K.: How is a data-driven approach better than random choice in label space division for multi-label classification? *Entropy* **18**, 282 (2016)
12. Tsoumakas, G., Katakis, I.: Multi label classification: an overview. *Int. J. Data Warehouse Min.* **3**(3), 1–13 (2007)
13. Tsoumakas, G., Katakis, I., Vlahavas, I.: Effective and efficient multilabel classification in domains with large number of labels. In: *Proceedings of the ECML/PKDD Workshop on Mining Multidimensional Data*, pp. 30–44 (2008)

14. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multi-label data. In: Maimon, O., Rokach, L. (eds.) *Data Mining and Knowledge Discovery Handbook*, pp. 667–685. Springer, Boston (2010). https://doi.org/10.1007/978-0-387-09823-4_34
15. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. *Mach. Learn.* **73**(2), 185–214 (2008)
16. Verikas, A., Gelzinis, A., Bacauskiene, M.: Mining data with random forests: a survey and results of new tests. *Pattern Recogn.* **44**(2), 330–349 (2011)



Infinite Mixtures of Markov Chains

Jan Reubold¹(✉), Ahcène Boubekki², Thorsten Strufe¹, and Ulf Brefeld²

¹ TU Dresden, Dresden, Germany

{jan.reubold, thorsten.strufe}@tu-dresden.de

² Leuphana University, Lüneburg, Germany

{boubekki, brefeld}@leuphana.de

Abstract. Facilitating a satisfying user experience requires a detailed understanding of user behavior and intentions. The key is to leverage observations of activities, usually the clicks performed on Web pages. A common approach is to transform user sessions into Markov chains and analyze them using mixture models. However, model selection and interpretability of the results are often limiting factors. As a remedy, we present a Bayesian nonparametric approach to group user sessions and devise behavioral patterns. Empirical results on a social network and an electronic text book show that our approach reliably identifies underlying behavioral patterns and proves more robust than baseline competitors.

1 Introduction

Being able to translate a user's behavior into an educated guess of her intent is often the key to provide a satisfying user experience. Users express different behavior in different contexts to satisfy their needs, fulfill a task, etc. [1]. Characteristic behavioral traits may thus serve as indicators for future behavior and capturing these traits is important in many application domains:

Content providers on the Web often rely on repeated user visits. Their success depends highly on how well they are able to anticipate a user's needs by providing the right content, at the right time, and in the right place. Accurately modeling user behavior not only predicts a user's actions but informs design and content decisions. This includes predicting what links a user will click on, deciding where webpage components should be placed, and what content to provide.

A similar problem arises in emerging areas such as *educational research* that aim to provide tailored learning environments and tutoring systems to children and students. Often it is either undesirable or not possible to build personalized models, and even when available, such models suffer from the cold start problem, or are unable to deal with context-dependent variations in user behavior. Accurately modeling user behavior leads to accurate assessments of a user's competency and allows for selecting next items, appropriate feedback, etc.

Recently, user behavior plays an increasing role in *security* related areas. Behavioral models are studied as replacements for passwords and intelligent pieces of operating systems are being developed to actively block security related components, such as access to a company data base, when the user is checking

news on Facebook. Similarly, security relevant features can be blocked by such a system if the user behavior deviates from the expected behavior; e.g., to prevent hacking a stolen device.

Traditionally, Markov models are frequently studied methods in behavioral contexts [3, 6, 7, 22] due to their good interpretability. The underlying idea is to exploit the sequential nature of user behavior and translate user sessions into Markov processes. Using Expectation-Maximization (EM)-based approaches [23], similar sessions can be grouped to draw conclusions about different types of users and their behaviors from the arising clusters. While there is nothing wrong with the general blueprint of these analyzes, they often suffer from being parametric approaches and using greedy optimization strategies that may lead to poor local optima. The problem arises because the optimal number of clusters is *a priori* unknown and needs to be identified with heuristics (e.g., [24, 25]) or trial and error. Often, this leads to repeated parameter estimations on subsets of the data. In addition, EM-based algorithms potentially converge to local optima and, therefore, several repetitions of the same experiment with random initializations are required. In the presence of today's data set sizes, the multiplicative consequences of deploying heuristics with EM-based algorithms quickly become prohibitive.

We present a non-parametric Bayesian approach to fit a mixture model of Markov chains to sequential data, turning behavior into data. We draw conclusions from the resulting models that constitute novel insights and show how these insights impact future developments and design decisions.

2 Related Work

Modeling user behavior is often performed using probabilistic models in combination with some sort of clustering. The most commonly studied type of approaches are based on Markov models [3, 6–9]. Early work investigates the use of probabilistic methods and subsequent publications use the formalism of Markov chains [6, 7] to build stochastic models that capture behavioral patterns. [3] further explores the idea by proposing a mixture model of Markov chains to divide data into meaningful groups and focus on these groups in the analysis. Here, each manifestation of a common behavioral pattern is represented by a Markov chain. Putting this in the context of our application scenario, a user interacts with a system and, by doing so, transitions between (possibly latent) states of a Markov model. Each state represents a possible interaction between user and system. Due to the use of first-order Markov chains, the next state is only conditioned on the previous state. The approach by [3] yields interpretable results and is computationally efficient. However, model selection may lead to sub-optimal results as identifying the number of groups is not always straight forward and the non-convex problem may have many local optima. A similar approach using a mixture model of hidden Markov models [8] takes intertwined click traces into account while [9] propose selective Markov models for identifying user behavior patterns.

Generally, higher-order Markov models [10–12] capture user behavior in more detail. However, [10] suffers from inefficient computations and results that are difficult to interpret. Other approaches require unreasonably large data sets as the model parameters grow exponentially with the number of states N and order o [11, 12]. [13, 14, 16] make use of Bayesian nonparametric mechanisms to control the complexity of the respective models. E.g., combining a temporal point process with a Bayesian nonparametric prior, [16] study the relation between both areas. The resulting Dirichlet-Hawkes process allows to model user behavior in greater detail compared to first-order Markov models. However, point processes focus on predictive performance and often lack interpretability.

To satisfy all requirements, we propose an approach that combines both, Bayesian nonparametric methods and Markov models. We derive a model that adapts to the complexity of the data and, at the same time, retains interpretability.

3 Non-parametric Bayesian User Behavior Models

In this section, we briefly introduce mixtures of Markov chains models and discuss their properties. After pointing out the drawbacks of this approach, we present a Bayesian nonparametric interpretation that mitigates these issues.

3.1 Mixtures of Markov Chains

Markov chains are probabilistic models for generating sequences of discrete events. The probability of observing an element directly depends on the previous one¹. Let us consider N sequences (or user sessions) $\mathbf{x}^{(i)} = (x_1^{(i)}, \dots, x_{T^{(i)}}^{(i)})$ of length $T^{(i)}$ over an alphabet M such that every $x_t^{(i)} \in M$ with $i \in \{1, \dots, N\}$. For ease of notation, every sequence is augmented by auxiliary start $x_0^{(i)} = S$ and terminal $x_{T^{(i)}+1} = E$ symbols, where $M \cap \{S, E\} = \emptyset$. The probability of observing adjacent elements is then given by the conditional $\theta_{u,v} = p(x_{t+1} = v | x_t = u)$ where $u \in M \cup \{S\}$ and $v \in M \cup \{E\}$. Note that the first event of a sequence is selected according to the prior surrogate $\theta_{S,v} = p(x_1 = v | S)$. Thus, the auxiliary start and terminal symbols allow for capturing prior and terminal distributions, respectively, where the latter eventually serves as a natural duration model of a cluster. The parameters θ are estimated by a maximum likelihood approach [3].

If there are several, say K , generating distributions instead of a single one, a mixture model of Markov Chains (MMC) is required for parameter estimation. Latent indicator variables z_i assign sequences to one of the K clusters and priors $\pi_k = p(z^{(i)} = k | \Theta)$ assess the importance of these clusters where $\Theta = (\pi_1, \dots, \pi_K, \theta_1, \dots, \theta_K)$. The quantity $p(z^{(i)} = k | \mathbf{x}^{(i)}, \Theta)$ estimates the probability that sequence i has been generated by the k -th component. To not clutter

¹ We focus on first-order dependencies but the approach is easily generalized to higher-order models; notation is quickly getting messy though.

the notations unnecessarily, we omit superscript i whenever context allows. The likelihood of the model is given by

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^K p(z = k|\Theta) \prod_{t=1}^{T+1} p(x_t|x_{t-1}, z = k, \Theta) = \sum_{k=1}^K \pi_k \prod_{t=1}^{T+1} \theta_{x_{t-1}, x_t}^k$$

Parameters Θ are estimated using Expectation Maximization (EM) and related techniques [3, 23].

While EM-based approaches yield interpretable results in an efficient and straight forward way, they suffer from two major drawbacks. Firstly, the actual number of components is generally unknown and consequently K becomes a parameter that has to be adjusted in the model selection. Secondly, the greedy inference by EM-based approaches can converge to local optima. This not only renders a single solution unquantifiable but, also implies repetitions of the same experiment necessary (e.g., using different initializations). Combining the two arguments leads to complex experimentations and quickly becomes tedious.

By contrast, our contribution addresses both limitations of EM-based approaches. Being a Bayesian nonparametric interpretation of the mixture of Markov chains, the number of components is adjusted in a data-driven way during the optimization. The latter is performed by a Gibbs sampling approach that does not share the greedy nature of EM-based methods.

3.2 Infinite Mixtures of Markov Chains

Our contribution, infinite Mixtures of Markov Chains (iMMC), makes use of a computationally efficient approximation to the hierarchical Dirichlet processes (HDP) [2], known as the degree L weak limit approximation [4]. The limiter L denotes the maximum cardinality of the approximated distribution. The approach encourages the learning of models with a state space of less than L components while allowing for the creation of new ones. It can be shown that such an approximation converges to the original HDP as $L \rightarrow \infty$ and provides a common solution to efficient Bayesian nonparametrics [20].

Graphical Model. Our model consists of a maximum number of L clusters, each comprised of a subset of events $M_l \subseteq M$ with $l \in \{1, \dots, L\}$. As before, we differentiate between observations \mathbf{x} and latent variables \mathbf{z} that assign sequences to clusters. The model is build of two well-known concepts in Bayesian nonparametrics, the Dirichlet distribution (Dir) and the finite-dimensional hierarchical Dirichlet process [2, 4]. A hierarchical Dirichlet process (HDP) consists of a two-layer hierarchy of Dirichlet processes (DP).

While the Dirichlet distribution is used to substitute the Multinomial distribution of the MMC to allow for an adaptive prior distribution over the cardinality of the clusters, the observation layer is modeled by a degree L weak limit approximation [4] which captures the Markovian structure of a cluster. The idea of this design choice is that the distribution over the events of a cluster serves

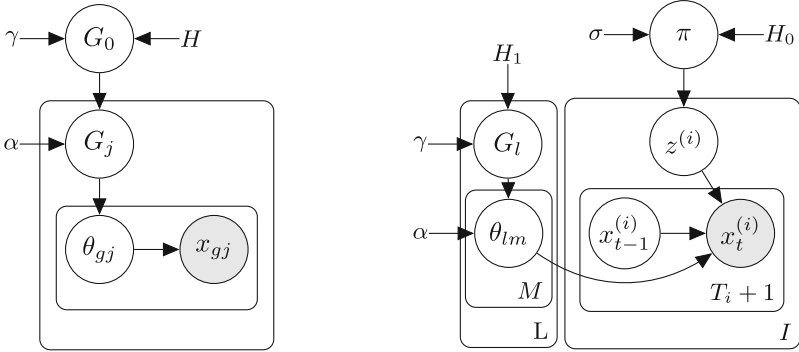


Fig. 1. (left) Graphical model of an HDP mixture model; (right) graphical model of the proposed iMMC; M is the set of events; I is the cardinality of the set of input sequences with T_i as the length of the corresponding sequence; $i \in I$ and $t \in \{0, \dots, T_i + 1\}$; white and gray nodes represent hidden states and observed states, respectively.

as a natural base measure to the emission distributions of the events. Here, the emission distributions denote the transition probabilities from a state to any other. By representing these emission distributions by DPs themselves, we build an HDP representing a cluster. Note that this way we define the Markov models by the emission distributions of its states.

The approximated HDPs consist of a Dirichlet G_l to model the state distribution within a cluster l and a set of subordinate Dirichlet distributions θ_{lm} , which represent the transitions within a cluster, i.e., the transition distribution given the current cluster l and its current state $m \in M_l$. The prior distributions π , G_l and θ_{lm} are then computed by

$$\begin{aligned} \pi | \sigma &\sim \text{Dir}(\alpha/L, \dots, \alpha/L) \\ G_l | \gamma &\sim \text{Dir}(\gamma/L, \dots, \gamma/L) \\ \theta_{lm} | \alpha, G_l &\sim \text{Dir}(\alpha G_{l1}, \dots, \alpha G_{lL}). \end{aligned} \quad (1)$$

Note that the prior and terminal state distributions are encoded within θ due to the augmentation of start and terminal symbols. Figure 1(right) shows the graphical model and the generative process of a single sequence based on the prior distributions is given by

$$z | \pi \sim \pi \quad x_t | z, x_{t-1} \sim \theta_{z x_{t-1}} \quad t \in \{1, \dots, T_i + 1\}. \quad (2)$$

Inference. To estimate parameters we make use of a two-step sampling algorithm which consists of the alternation of sequence assignments and parameter updates. In the assignment phase we obtain a realization of the latent parameters which is then used for the update of the prior distributions. These two steps are then repeatedly run to obtain the final model parameters. In the following we explain both steps in detail.

Algorithm 1. Blocked Gibbs sampler for iMMC

Given the hyperparameters σ, γ, α

(i) Initialize prior distributions according to Eq. 1

Until convergence **do**:

(ii) Obtain a realization of z according to Eq. 5

(iii) During assignment step update auxiliary variables as follows:

→ For each assigned sequence, increment:

· $b_{l=z^{(i)}} \equiv \#$ observations assigned to cluster l

→ For each observation in the sequence, increment:

· $d_{l=z^{(i)}, x_t} \equiv \#$ observations of state x_t assigned to $l = z^{(i)}$

· $s_{l=z^{(i)}, x_{t-1}, x_t} \equiv \#$ transitions from x_{t-1} to x_t in $l = z^{(i)}$

(iv) Re-sample prior distributions

(v) Build final model from multiple sample-sets of the parameters

Assignment Step. Given randomly initialized prior distributions (see Eq. 1), we compute the likelihood of a sequence x as

$$p(\mathbf{x}|\Theta) = \sum_{l=1}^L p(z = l|\Theta) \prod_{t=1}^{T+1} p(x_t|x_{t-1}, z = l, \Theta) = \sum_{l=1}^L \pi(l) \prod_{t=1}^{T+1} \theta_{lx_{t-1}}(x_t), \quad (3)$$

where x_0 and x_{T+1} represent the artificial boundary nodes and π the prior distribution over the clusters. The marginal distribution is

$$p(x|z = l, \Theta) \propto \pi(l) \prod_{t=1}^{T+1} \theta_{lx_{t-1}}(x_t). \quad (4)$$

Therefore, the assignments can be sampled as

$$z^{(i)} \sim \text{Mu} \left(\sum_{l \in L} p(x|z = l, \Theta) \delta_l \right), \quad (5)$$

where δ represents the Dirac delta.

Update Step. After obtaining a new sample of assignments the prior distributions have to be updated. This is an essential step in the Gibbs sampler and, in our case, straight-forward given that all distributions consist of DPs. Therefore, statistics are gathered during the assignment step. We keep track of the state distribution and transitions within the clusters. Thus, $d_{l,m}$ records the number of observations of state m assigned to cluster l and s_{l,m_1,m_2} records the number of transitions from state m_1 to state m_2 within cluster l . Finally, b_l keeps track of the number of observations assigned to luster l . For each iteration, the auxiliary variables document the assignment step. Then, we can re-sample the

distributions using the statistics as the new evidence. A summary of the entire inference process is given in Algorithm 1. Note, that, while seemingly similar to classic EM-approaches, the Gibbs sampler is based on sampling rather than on ML solutions. Therefore, it can be shown, that under certain conditions the sampler will converge to the global optimum [26].

4 Experiments

We first evaluate the clustering performance of our model in controlled scenarios to understand its effectiveness and to shed light on extreme cases. The synthetic nature of the data allows us to accurately evaluate the clustering performance of our approach. Then we will focus on the interpretability of the clusters and of the extracted patterns. Therefore, we extract usage patterns of users browsing a social network website without prior knowledge. Finally, an analysis of the behavior on an electronic textbook using iMMC will show that the obtained patterns correlate with the success of the corresponding student, suggesting that behavior patterns hold further, sensitive information about students.

4.1 Synthetic Data

In this section we compare the clustering performance of our algorithm, the infinite mixture model of Markov chains (iMMC), to the traditional mixture model of Markov chains approach (MMC). We pick the latent Dirichlet allocation (LDA) [27] as an additional baseline to assess the importance of the sequential information contained in the observations. LDA only makes use of the frequency count of events within a sequence.

We generate three synthetic scenarios to generate different sets of clusters. In the context of user behavior, a cluster represents the causal reason for an observed sequence of events: clusters thus serve as proxies for user intention/interest. Their state spaces are the set of events that are associated with one or more clusters.

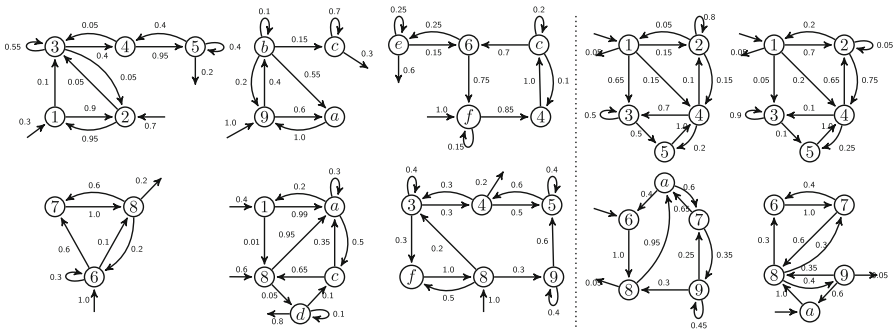


Fig. 2. Generative processes of scenario II (left) and scenario III (right); states are indexed by hexadecimal numbers (1-f).

A learning task is simpler when state spaces are disjoint (Scenario I). An example are clusters like ‘cooking’ and ‘driving a car’ that have no state spaces of events in common. Learning tasks with fully overlapping state spaces are more difficult (Scenario III, Fig. 2(right)). Examples are clusters that share many events such as ‘cooking’ and ‘baking’ or ‘driving a car’ and ‘driving a motorcycle’. The learning task in Scenario II (Fig. 2(left)) addresses both characteristics.

Given a scenario, we obtain a corresponding data set by selecting uniformly at random one of its clusters. Then we run its generating process which yields a sequence of actions. This procedure is repeated until we have the desired number of actions in the set of generated sequences. For each scenario we evaluate the algorithms on data sets of sizes of 10,000, 100,000 and 250,000 data points. For each combination of scenario and data set size, we generate 10 data sets and report on results of the averaged performances over 5 runs for each of these data sets. While we use a single set of hyperparameter values for our algorithm (each is set to 1), we supply the MMC with the correct number of clusters and apply a soft clustering. For LDA we transform each sequence into a frequency vector of events occurring in the sequence.

Table 1. Error rates for the synthetic clustering tasks; each data set consists of 10k, 100k, and 250k data points (small, medium, large).

	Scenario I			Scenario II			Scenario III		
	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large
LDA	20.92%	28.14%	28.62%	14.69%	12.09%	20.20%	27.95%	29.54%	29.06%
MMC	19.60%	9.90%	5.13%	5.94%	6.78%	4.77%	14.26%	20.36%	8.47%
iMMC	0.14%	2.23%	0.26%	0.00%	0.54%	2.78%	8.61%	5.82%	5.15%

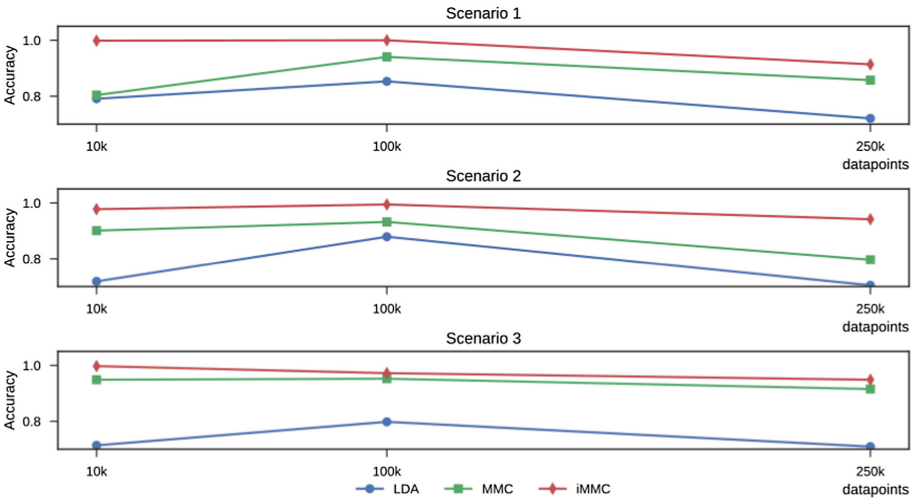


Fig. 3. Accuracy of each method on different scenarios and for dataset sizes.

Even though MMC was provided with the correct number of clusters and our algorithm had to adjust it to the data, our algorithm is as efficient as MMC.

Table 1 and Fig. 3 shows the overall clustering performance of both algorithms on all data sets and scenarios. In all cases, our algorithm outperforms MMC.

4.2 Facebook Data

In this experiment, we demonstrate how the model can be applied for information extraction tasks from huge dataset. This is especially useful for tasks that come with no or only little prior knowledge. The data set for the next evaluation contains user navigation data from Facebook [15]. For each user, the invoked pages are recorded and grouped into sessions. Examples for such invoked pages are ‘Login’, ‘Newsfeed’, ‘Load more news’, ‘Like’, etc.. The dataset contains 152 unique invoked pages, 49,479 sessions of 2,749 users, and 8,197,308 observations. Every session is interpreted as a sequence of observations.

The most frequently observed behavioral pattern is user’s checking for updates on the newsfeed by *waking up* the device and, without performing any additional activity, *put to sleep* shortly after. Figure 4 depicts two more complex patterns of users on Facebook. The first pattern, on the left, describes passive user behavior without any direct communication. Users following this patterns tend to look at their newsfeed (*News*) or at their own timeline (*ownTL*). While updating (represented by the loop on *ownTL*) or scrolling (*moreTL*) their own timeline, they would sometimes be interested in someone else’s time-line (*otherTL*). There, they scroll through it but will most likely go back to their own timeline. They tend to look at more entries (*viewNews*) from their newsfeed and interact (self-loop) with them. If they open a gallery (*Photo*), they would look at several pictures (self-loop on *Photo*) before returning to their previous activity.

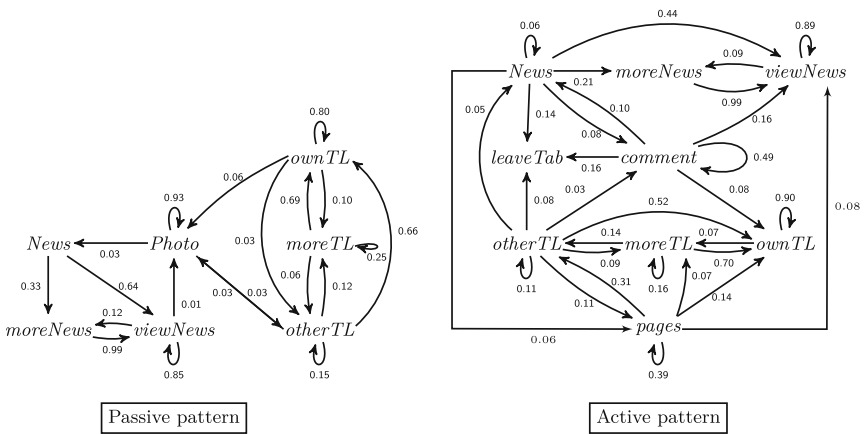


Fig. 4. An exemplary solution of the identified clusters; exit states are omitted, their probability equals 1 minus the sum of emission probabilities of a state.

The pattern in the right part of Fig. 4 describes a more active behavior. While also browsing their Facebook universe, users frequently *comment* on news-feeds' and timelines' entries. Additionally, the users visit fan and company pages (*pages*). The iMMC algorithm successfully distinguished different session behaviors without any prior knowledge on the data, nor dependencies between events.

4.3 Electronic Text Books

In this section, we present insights on the usage patterns of students interacting with an electronic text book for history called the mBook [5]. We show that identified usage patterns correlate with psychometric scores.

Among others, the mBook has been successfully deployed in the German-speaking community of Belgium. Together with psychologists and didacticians, we aim to evaluate the pros and cons of daily use in classrooms on children and teachers. In addition to an event log that tracks all user actions in the book, demographic variables as well as variables measuring competencies and interest are regularly assessed. Since 2013, about 3,000 users have created 370,000 sessions. In this experiment, we focus on 803 sessions of a subset of 286 users between February and March 2017. Our aim is to identify characteristic usage patterns to later search for correlation with psychometric variables.

Related studies reveal that time-on-page and cursor trajectories often serve as indicators for student engagement [18,19]. However, in our case, the text book is mainly used on tablets in class rooms and, hence, cursors or eye tracking are not available. We thus aim to identify alternative indicators that are precise enough to capture characteristic traits of different behavior. We define and differentiate 75 atomic events that a user can trigger, ranging from pressing a button to various scrolling performances. The latter are further divided into 9 events: *scroll.direction.duration*. The direction can be *up*, *down* or *fix* if the movement is of less than 10 pixels. The duration can be *fast*, *medium* or *slow* for event duration of respectively less than 1 s, between 1 and 3 s and more than 3 s. In the following, node names will be abbreviated using only the first letter. For example a *scroll.down.fast* is reduced to *d.f*.

In contrast to the analysis of the Facebook data set, where the huge amount of data allowed for a deployment of MMC, in this case a MMC would fail due to the lack of a sufficient amount of data; information criteria are known to perform poorly when the sample size is smaller than the number of parameters [17] as shown in Fig. 5(left). The evolution of three information criteria AIC [24], AICc [21], and BIC [25] is depicted for different numbers of clusters where every point in the figure denotes the best result out of 30 repetitions. Theoretically, the minima of these curves are supposed to give the optimal solutions given the involved parameters. Due to the ill-posed optimization problem, however, the criteria grow almost linearly. The AIC curves reaches a minimum for two clusters, what is not really interesting. Thus information criteria do not allow to draw conclusion.

By contrast, our Bayesian approach successfully clusters the data using $\gamma = 2$, $\sigma = 1.5$, $\lambda = 2.4$, $L = 100$ and 10,000 iterations. After every 1,000 iterations, an

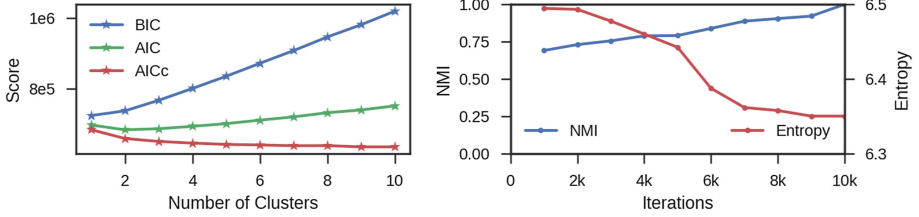


Fig. 5. Left: BIC, AIC and AICc for MMC. Right: NMI and entropy for iMMC (Color figure online)

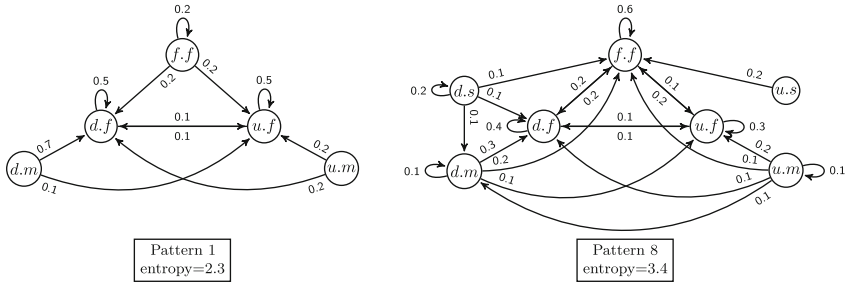


Fig. 6. Two exemplary scrolling patterns.

intermediate clustering is computed as the average of the last 1,000 iterations. The first intermediate clustering is based on 34 clusters, the final solution settles on 32 clusters. The evolution of the solution is shown in Fig. 5(right). The blue line (left scale) represents the evolution of the normalized mutual information (NMI) relative to the final solution. The red line (right scale) refers to the entropy of the clustering for the actual iteration. After 7,000 iterations the NMI indicates that the clustering is already 90% similar to the final one. The decrease in entropy shows that the algorithm merges the data into fewer clusters. The plateau after 7,000 iterations indicates fine granular changes of cluster memberships.

There are eight resulting clusters with at least 20 sessions. We focus on the scrolling events and show two patterns in Figure 6 realizing the smallest and highest entropy, respectively. Note that the weights do not sum up to one, as we ignore outgoing edges to non-scroll events in this analysis.

The first thing to notice is that in Pattern 1, *scroll.fix.** cannot be reached from another type of scroll. Either it starts a scrolling sequence or it indicates mis-usage or hesitation of the user. Although Pattern 8 is more complex, it shares the fact that users tend to not transit to slower scrolls. This can be interpreted by the observed behavior that ‘longer’ scrolls are corrected by faster ones. This is typical behavior for users who are scrolling while reading the text on the page. This is also reflected in high self-transition probabilities of *scroll.down.slow* and *scroll.fix.fast*. Multiple ways to reach this last event are likely caused by stopping a scroll with a small scroll and keeping the finger on the tablet.

Psychometric Correlations. During the four years of the experiment, the children are assessed at the end of each school year. Five factors are measured. Competency and knowledge in the field are assessed using item response theory [28,29]. Additionally, their motivation, access to digital devices and their skills in the usage of these are assessed by multiple choice questionnaires (advanced skills weight more than simple ones).

To correlate the assessed variables with our clustering, we represent clusters by the average score of all children who have sessions in the cluster. We compute Pearson correlation coefficients [21] that are adjusted for small sample sizes for the 81 possible transition probabilities between scroll events and the 8 resulting clusters with at least 20 elements.

The maximum and minimum correlations for the assessed variables are reported in Table 2. Except for motivation, high correlated transitions for every variable end with a *scroll.up.fast* and a change in direction. Knowledge has a correlation of almost 1.0 with *scroll.down.medium* \rightarrow *scroll.up.fast*, and of nearly -1.0 with *scroll.down.slow* \rightarrow *scroll.down.medium*. Pattern 8 is the only pattern containing these two edges. However, the correlations cancel out in the final result. Figure 7 confirms that cluster 8 loads only weakly on knowledge compared to the others.

The first row in Fig. 7 shows the loadings for the 8 biggest clusters. The clusters are organized from top to bottom according to their entropy.

Patterns 1 and 8 (see Fig. 6) are extracted from clusters 1 and 8, respectively. Both patterns are often observed by pupils with high competencies in history. Therefore, these patterns may serve as behavioral indicators for a user’s competency. This finding is supported by the high correlation of cluster 1 with the prior knowledge of the user. Seemingly, knowledgeable children prefer simpler scrolling patterns. By contrast, cluster 2 contains highly motivated children that possess high computer skills. The pupils in cluster 6 are also motivated but do not possess such a high ICT literacy and thus do not know to handle electronic devices that well.

The second row in Fig. 7 displays the values among clusters of the most strongly correlated transitions to the corresponding score. Negative correlations are not shown for interpretability. These plots give an impression of the correlations. For knowledge and motivation scores, the probability of *scroll.down.medium* \rightarrow *scroll.up.fast* and *scroll.fix.fast* \rightarrow *scroll.fix.fast* could

Table 2. The most strongly correlated event transitions for each score.

Score	Max corr.	Event	Min corr.	Event
Competence	0.697	<i>f.f</i> \rightarrow <i>u.f</i>	-0.719	<i>u.m</i> \rightarrow <i>u.s</i>
Knowledge	0.962	<i>d.m</i> \rightarrow <i>u.f</i>	-0.947	<i>d.s</i> \rightarrow <i>d.m</i>
Motivation	0.748	<i>f.f</i> \rightarrow <i>f.f</i>	-0.714	<i>f.f</i> \rightarrow <i>u.f</i>
IT access	0.751	<i>d.s</i> \rightarrow <i>u.f</i>	-0.735	<i>f.f</i> \rightarrow <i>d.f</i>
IT skill	0.837	<i>d.s</i> \rightarrow <i>u.f</i>	-0.743	<i>d.m</i> \rightarrow <i>d.s</i>

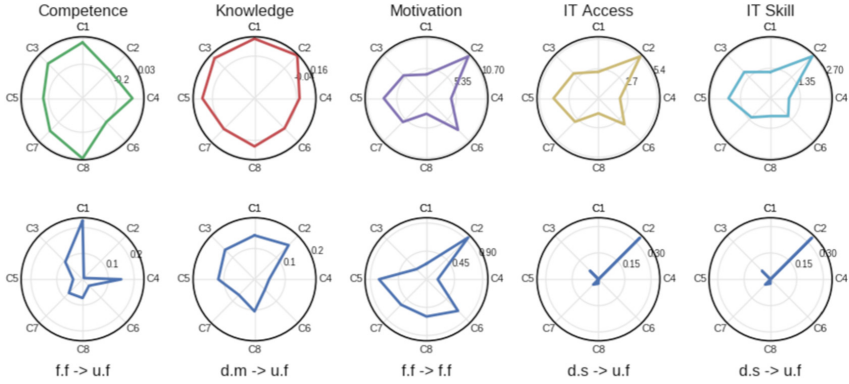


Fig. 7. Scores and probabilities of their most correlated transition for the 8 biggest clusters.

be used to predict their respective scores in the assessment. With respect to competence, a high transition probability seemingly also implies a high score in the assessment. However the opposite does not hold true. Cluster 8, as also seen in Fig. 6, has a smaller probability of transitioning from *scroll.fix.fast* to *scroll.up.fast*, although the average competency score of the cluster is the largest.

Our results show for the first time that behavioral indicators in electronic text books can be identified to discriminate between children. Results like this will have a high impact on the next generations of electronic text books so that they become adaptive and provide individual learning environments for every child.

5 Conclusion

We presented a Bayesian nonparametric approach to modeling user behavior. The nonparametric nature of our approach allowed for the efficient identification of the underlying clusters within user event data. Our model showed significant improvements over related approaches when analyzing such data. We obtained a natural state-duration model by capturing end-state distributions of the clusters. The models allowed us to capture state durations based on the dynamics of the cluster. Furthermore, representing each cluster as a Markov chain led to easily interpretable results that may impact design decisions and future developments of the respective service.

Acknowledgements. This research has been funded in parts by the German Science Foundation DFG under grant GRK/1907 and by the German Federal Ministry of Education and Science BMBF under grant QQM/01LSA1503C.

References

1. Mitchell, A., Olmstead, K., Purcell, K., Rainie, L., Rosenstiel, T.: Understanding the participatory news consumer (2010)
2. Teh, Y.W., Jordan, M.I., Beal, M.J., Blei, D.M.: Hierarchical dirichlet processes. *J. Am. Stat. Assoc.* **101**(476), 1566–1581 (2006)
3. Cadez, I., Heckerman, D., Meek, C., Smyth, P., White, S.: Visualization of navigation patterns on a web site using model-based clustering. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 280–284 (2000)
4. Ishwaran, H., Zarepour, M.: Exact and approximate sum representations for the dirichlet process. *Can. J. Statistics/La Revue Canadienne de Statistique* **30**(2), 269–283 (2002)
5. Schreiber, W., Sochatzy, F., Ventzke, M.: Das multimediale Schulbuch - kompetenzorientiert, individualisierbar und konstruktionstransparent. In: Analyse von Schulbüchern als Grundlage empirischer Geschichtsdidaktik, pp. 212–232 (2013)
6. Pirolli, P.L., Pitkow, J.E.: Distributions of surfers' paths through the world wide web: empirical characterizations. *World Wide Web* **2**(1–2), 29–45 (1999)
7. Manavoglu, E., Pavlov, D., Giles, C. L.: Probabilistic user behavior models. In: Third IEEE International Conference on Data Mining, ICDM 2003. IEEE (2003)
8. Ypma, A., Heskes, T.: Automatic categorization of web pages and user clustering with mixtures of hidden markov models. In: Zaïane, O.R., Srivastava, J., Spiliopoulou, M., Masand, B. (eds.) WebKDD 2002. LNCS (LNAI), vol. 2703, pp. 35–49. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39663-5_3
9. Deshpande, M., Karypis, G.: Selective markov models for predicting web page accesses. *ACM Trans. Internet Technol. (TOIT)* **4**(2), 163–184 (2004)
10. Mochihashi, D., Sumita, E.: The infinite markov model. In: NIPS, pp. 1017–1024 (2007)
11. Bühlmann, P., Wyner, A.J.: Variable length markov chains. *Ann. Stat.* **27**(2), 480–513 (1999)
12. Begleiter, R., El-Yaniv, R., Yona, G.: On prediction using variable order markov models. *J. Artif. Intell. Res.* **22**, 385–421 (2004)
13. Dubey, A., Hwang, S., Rangel, C., Rasmussen, C.E., Ghahramani, Z., Wild, D.L.: Clustering protein sequence and structure space with infinite gaussian mixture models. In: Pacific Symposium on Biocomputing, pp. 399–410 (2003)
14. Brown, D.P.: Efficient functional clustering of protein sequences using the dirichlet process. *Bioinformatics* **24**(16), 1765–1771 (2008)
15. Paul, T., Puscher, D., Strufe, T.: Improving the Usability of Privacy Settings in Facebook. *CoRR* (2011)
16. Du, N., Farajtabar, M., Ahmed, A., Smola, A.J., Song, L.: Dirichlet-Hawkes processes with applications to clustering continuous-time document streams. In: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 219–228 (2015)
17. Giraud, C.: Introduction to High-dimensional Statistics, vol. 138. CRC Press, Boca Raton (2014)
18. Cocea, M., Weibelzahl, S.: Cross-system validation of engagement prediction from log files. In: Duval, E., Klamma, R., Wolpers, M. (eds.) EC-TEL 2007. LNCS, vol. 4753, pp. 14–25. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75195-3_2

19. Salmeron-Majadas, S., Santos, O.C., Boticario, J.G.: Exploring indicators from keyboard and mouse interactions to predict the user affective state. In: Educational Data Mining (2014)
20. Kurihara, K., Welling, M., Teh, Y.W.: Collapsed variational dirichlet process mixture models. In: IJCAI 2007, pp. 2796–2801 (2007)
21. Olkin, I., Pratt, J.W.: Unbiased estimation of certain correlation coefficients. *Ann. Math. Stat.* **29**(1), 201–211 (1958)
22. Haider, P., Chiarandini, L., Brefeld, B.: Discriminative clustering for market segmentation. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2012)
23. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. Ser. B (methodological)* **39**(1), 1–38 (1977)
24. Akaike, H.: A new look at the statistical model identification. *IEEE Trans. Autom. control* **19**(6), 716–723 (1974)
25. Schwarz, G.: Estimating the dimension of a model. *Ann. Stat.* **6**(2), 461–464 (1978)
26. Roberts, G.O., Smith, A.: Simple conditions for the convergence of the gibbs sampler and metropolis-hastings algorithms. *Stoch. Processes Appl.* **49**(2), 207–216 (1994)
27. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
28. Baker, F.B.: The basics of item response theory (2001). For full text: <http://ericae.net/irt/baker>
29. DeMars, C.: Item Response Theory. Oxford University Press, New York (2010)



Community-Based Semantic Subgroup Discovery

Blaž Škrlj¹(✉), Jan Kralj², Anže Vavpetič², and Nada Lavrač^{2,3}

¹ Jožef Stefan International Postgraduate School,
Jamova 39, 1000 Ljubljana, Slovenia
`blaz.skrlj@ijs.si`

² Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
{`jan.kralj, anze.vavpetic, nada.lavrac`}@ijs.si

³ University of Nova Gorica, Vipavska 13, 5000 Nova Gorica, Slovenia

Abstract. Modern data mining algorithms frequently need to address learning from heterogeneous data and knowledge sources, including ontologies. A data mining task in which ontologies are used as background knowledge is referred to as semantic data mining. A special form of semantic data mining is semantic subgroup discovery, where ontology terms are used in subgroup describing rules. We propose to enhance ontology-based subgroup identification by Community-Based Semantic Subgroup Discovery (CBSSD), taking into account also the structural properties of complex networks related to the studied phenomenon. The application of the developed CBSSD approach is demonstrated on two use cases from the field of molecular biology.

Keywords: Semantic data mining · Bioinformatics
Community detection · Network analysis · Term enrichment analysis

1 Introduction

Modern machine learning approaches are capable of using continuously increasing amounts of information to explain complex phenomena in numerous fields, including biology, sociology, mechanics and electrical engineering. As there can be many distinct types of data associated with a single phenomenon, novel approaches strive towards the integration of different, heterogeneous data and knowledge sources into unified predictive or descriptive models.

In such settings, prior knowledge can play an important role in the development and deployment of learning algorithms in real world scenarios. Background knowledge can come in many forms, which introduces additional complexity to the modeling process, yet can have a great impact on the model's performance. For example, Bayesian methods can be leveraged to incorporate knowledge about prior states of a system, i.e. prior distributions of random variables being modeled. In the Bayesian setting, the prior knowledge is incorporated via conditional probabilities used in the Bayes rule for posterior probability calculation.

For example, Bayesian methodology is in widespread use in the field of phylogenetics, where Bayesian inference is used for reconstruction of evolutionary trees [1]. A different modeling technique was used in a biological application by Madahian et al. [2], where a general linear model was developed to aid gene expression profiling, achieving better predictive accuracy by using prior knowledge based on the index rank of the term “cancer” in the underlying background knowledge. Background knowledge can be encoded also more explicitly, as an additional knowledge source to be used in learning the models. A machine learning discipline that relies heavily on the use of explicitly encoded background knowledge is inductive logic programming (ILP) [3]. In ILP, background knowledge is used along with the examples to derive hypotheses in the form of logic programs, which explain the positive examples.

Semantic Data Mining. Semantic data mining (SD) [4] is a field of machine learning that employs curated domain knowledge in the form of ontologies as background knowledge used in the learning process. An ontology can be represented as a data structure consisting of semantic triplets $T(S, P, O)$, which represent the subject, its predicate and the object. Resource Description Format (RDF) hypergraph is a data model commonly used to operate at the intersection of data and the ontologies. There are many existing approaches, which use background knowledge in the form of an ontology to obtain either more accurate or more general results. First, knowledge in the form of ontologies can represent constraints, specific to a domain. It has been empirically and theoretically demonstrated, that using background knowledge as a constraint can improve classification performance [5]. The RDF framework provides also the necessary formalism to leverage the graph-theoretic methods for ontology exploration. Random walks and large scale motif sampling are some of the techniques used to discover indirectly associated biological terms [6]. Semantic clustering is an emerging field, where semantic similarity measures are used to determine the clusters using the background knowledge, in a manner similar to, for example, k -means family of clustering algorithms. Semantic clustering is frequently used in the area of document clustering [7]. Large databases in the form of RDF triplets exist for many domains. For example, the Bio2RDF project [8] aims at integrating all major biological databases and joining them under a unified framework, which can be queried using SPARQLQ—a specialized query language. The BioMine methodology is another example of large-scale knowledge graph creation, where biological terms from many different databases are connected into a single knowledge graph with millions of nodes [9]. Despite such large amounts of data being freely accessible, there remain many new opportunities to fully exploit its potential for knowledge discovery.

Semantic Subgroup Discovery. Semantic subgroup discovery (SSD) [10,11] is a field of data mining named subgroup discovery, which is compliant with the paradigm of rule learning that expect a labeled training set, where class labels are used to denote the groups for which descriptive rules describing groups of

instances of interest are to be learned. Apart from experimental data, a semantic subgroup discovery algorithm leverages background knowledge in the form of ontologies in order to guide the rule learning process. For example, the Hedwig algorithm [10, 12] accepts the input in the form of ontologies and the instances, grouped into different classes, and the individual instances are mapped to the ontology terms, while rule learning is guided by the hierarchical relations between the considered ontology terms. Hedwig is capable of using an arbitrary ontology to identify latent relations explaining the discovered subgroups of instances.

Complex Networks. Complex networks are graphs with distinct, real world topological properties [13]. Many natural phenomena can be described using graphs. They can be used to model physical, biological, chemical and mechanical systems [14, 15]. Real world networks can be characterized with distinct statistical properties regarding their node degree distribution, component distribution or connectivity [16]. Complex biological and social networks are also known to include many communities, i.e. smaller, distinct units of a network [17]. Complex networks are commonly used in modeling systems, where extensive background knowledge is not necessarily accessible. Motif finding, community detection and similar methods can provide valuable insights into the latent organization of the observed network.

In this work we propose a methodology, where iteratively constructed complex networks are used to identify relevant subgroups, which are used as input for the process of semantic subgroup discovery. We demonstrate that new knowledge can be obtained using existing, freely accessible heterogeneous data in the form of complex networks and ontologies. In the next sections we present the proposed methodology and demonstrate the use of the new approach on two datasets from the life science domain, where the complementarity with existing enrichment analysis tools is demonstrated.

2 Methodology

This section presents the proposed approach to semantic subgroup discovery from complex networks, named CBSSD (Community-Based Semantic Subgroup Discovery). The proposed approach operates on a list of terms connected with the studied phenomenon. The main steps include: network construction, community detection and semantic subgroup discovery. The proposed methodology is depicted in Fig. 1.

Constructing the Network of Associations. A list of relevant biological terms is used to construct a term network. The network is constructed using the BioMine methodology [9]; individual terms are used as seeds for crawling the BioMine knowledge graph, which already includes millions of term associations across main biological databases, such as UniProt [18], Kegg [19], and GenBank [20]. The final knowledge graph G_f is constructed incrementally, by querying one

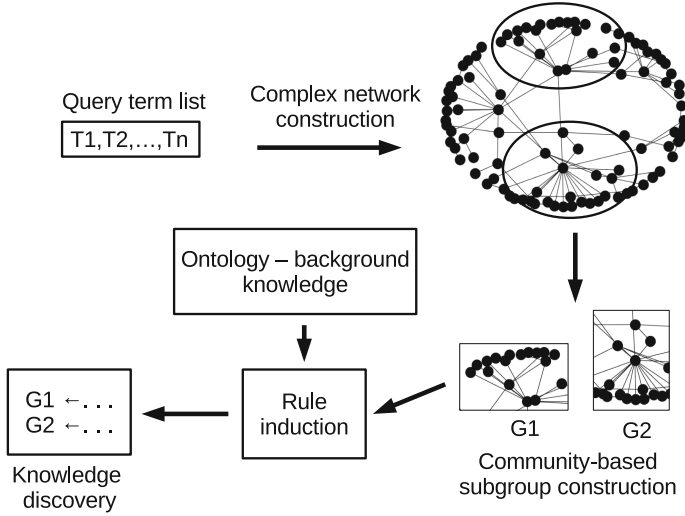


Fig. 1. Schematic representation of the proposed CBSSD procedure. Complex graph’s communities are used to identify possible subgroups in the input term list. The subgroups are further explained using semantic subgroup discovery with background knowledge.

term at a time. This knowledge graph consists of a set of graphs $\{G_1, \dots, G_n\}$, where n is the total number of query terms and, for each i , $G_i = (V_i, E_i)$. In order to obtain the final graph G_f , node and edge information from $\{G_1, \dots, G_n\}$ is joined into a single graph. Throughout the network construction process, nodes and edges can not be duplicated—once the node is present in the final graph, only new edges can be added. Final set of nodes V_f thus equals $\bigcup_{i=1}^n V_i$ and final set of edges E_f similarly equals $\bigcup_{i=1}^n E_i$.

Community Detection in Homogeneous Networks. Once the network is constructed, a network community detection algorithm is used to identify interesting subsets of the network, which are directly mapped to groups within the input query list. We use the Louvain algorithm [21], which is based on the network modularity measure [22] defined for splitting the network into two modules (m_i and m_j) as follows:

$$\xi = \frac{1}{2m} \sum_{i=1}^n \sum_{j=1}^n \left[A_{i,j} - \frac{d_i d_j}{2m} \right] \frac{m_i m_j + 1}{2} \tag{1}$$

where the ξ represents the modularity, m the number of all edges, A the adjacency matrix (i.e. $A_{i,j}$ is equal to 1 if the i -th and j -th node are connected and 0 if they are not), and d_i is the degree of node u_i . Term m_i represents a membership function, which returns 1 if a specific node is present in the observed module and -1 otherwise. The final community partition includes all the nodes. The Louvain

algorithm is one of the most scalable community detection methods due to its $\mathcal{O}(n \log(n))$ time complexity. For this step, the constructed knowledge graph was interpreted as an undirected graph, which is a feasible assumption as long as we are interested only in biological associations. The community detection procedure returns sets of nodes $\{C_{1\dots n}\}$ that represent individual communities. Each node in the network belongs to exactly one community (i.e. the communities are non-overlapping). We are interested in finding subgroup descriptions of these communities. In order to do this, each community C_i becomes a class label T_i . The terms from the input list are partitioned to individual classes according to the community they belong to. This way, input terms are grouped into distinct classes, yet no additional terms are added as they could introduce unnecessary noise in the semantic subgroup discovery step.

Community Detection in Heterogeneous Networks. As the networks under consideration consist of many distinct layers (node types), our methodology can also account for such organization without additional simplification of the network. For such tasks, we leverage the state-of-the-art InfoMap algorithm for multilayer community detection [23]. This algorithm’s objective is to minimize the information gain, formulated as the map equation:

$$L(M) = q \curvearrowright H(Q) + \sum_{i=1}^m p_{\circ}^i H(p^i) \quad (2)$$

where $L(M)$ represents the per-step description length for module partition M . For module partition M of n nodes into m modules, $L(M)$ is the lower bound of the average length of a *code word* describing the trace of a random walker. The partition resulting in the shortest description length is believed to best represent the network dynamics. The $q \curvearrowright H(Q)$ represents the total probability that the random walker enters any of the m modules under consideration. The entropy of the relative rates $H(Q)$ is used to measure the smallest average *code word* length that is theoretically possible. The p_{\circ}^i represents the total probability that any node in the module is visited, plus the probability that the random walker exits the module and the exit *code word* is used. Entropy $H(p^i)$ of the relative rates at which the random walker exits module i and visits each node in module i , measures the smallest average *code word* length that is theoretically possible. For completeness, our approach includes also a variant of the InfoMap algorithm, which detects communities in homogeneous networks, i.e. networks consisting of single node types.

Preparation of the Background Knowledge. Semantic rule learning requires the data to be encoded in the form of RDF triplets $T(S, P, O)$, where S is the subject, P the predicate and O the object. The experimental data from the previous step was converted into RDF triplets in accordance with Hedwig, the algorithm used in the rule discovery process [10]. Hedwig is capable of leveraging the background knowledge in the form of ontologies to guide the rule

construction process. It does so by using the hierarchical relations between the ontology terms. Rules are initially constructed using more general terms and further refined using more specific terms. As the CBSSD methodology is primarily developed for the field of bioinformatics, our main source of background knowledge in this study is the Gene Ontology (GO) [24] database, one of the largest semantic resources for biology. It includes tens of thousands of terms, which together form a directed acyclic graph, directly usable by SSD tools.

For Hedwig to perform rule construction, two conditions must be met. First, individual term names from the community detection step need to have the corresponding GO term mappings, and second, the whole gene ontology must be provided as a source of background knowledge. This requires that the discovered communities are encoded in the form of semantic triplets. Such encoding is achieved by treating each observed community as an individual target class, where all of its terms are considered as instances of this class. The key aspect of the rule generation procedure is the definition of the predicate, which will be used for finding suitable rule conjunctions. The objective function can thus be formulated as learning a rule set Δ for individual classes $\zeta_{1,\dots,n}$ using background knowledge (Ξ) in the form of ontologies, and class instance embeddings (in the semantic space) γ , such that the likelihood of individual class representations ζ_x for $x \in \{1, \dots, n\}$ is maximized, which is formulated as follows:

$$\Delta_{\zeta_1, \dots, \zeta_n} = \arg \max_{i \in \{1, \dots, n\}} \left[Pr(\Delta_{\zeta_i} | \Xi, \gamma) \right]. \quad (3)$$

By convention, we use the *subClassOf* predicate when constructing the knowledge base for the Hedwig algorithm. Individual rules' p -values are determined by the Fisher's exact test (FET), a non-parametric, contingency table-based procedure, where a difference in coverage between two rules is leveraged to select the better one. The FET test is based on the hypergeometric distribution, in which a random variable X is distributed as

$$Pr(X = k) = \frac{\binom{K}{k} \binom{N-K}{n-k}}{\binom{N}{n}} \quad (4)$$

where N is the number of all examples, K is total number of positive examples, n is the rule coverage (number of covered terms) and k is the number of covered positives in the context of a single rule or a beam. Further, multiple hypothesis correction (e.g., Bonferroni or Benjamini-Hochberg) is applied in order to reduce the false discovery rate.

Final Formulation of the CBSSD Approach. First, individual input terms are used to construct the heterogeneous network related to the studied phenomenon. Communities are identified (*CommunityDetection* step) and the input term list is partitioned according to the presence of individual terms within specific communities (*PartitionByCommunity*). Finally, background knowledge in the form of ontologies is used to construct meaningful representations of

individual partitions. The CBSSD approach can thus be further formalized as described in Algorithm 1.

Algorithm 1. Pseudocode of the CBSSD approach

Data: terms (T), ontologies (Ξ), mapping function (M), network generator (G)

Result: Enriched rule sets $\Delta_{\zeta_1, \dots, \zeta_n}$

$S_f := \emptyset$;

foreach $t \in T$ *processed* **do**

$S_{f_{edges}} := G(t)_{edges} \cap S_f$;

$S_{f_{nodes}} := G(t)_{nodes} \cap S_f$;

end

$C_{1\dots n} := \text{CommunityDetection}(S_f)$;

$P_{1\dots n} := M(\text{PartitionByCommunity}(T, C_{1\dots n}))$;

foreach $P_x \in P_{1\dots n}$ **do**

while Δ_{ζ_x} *not final* **do**

$\Delta_{\zeta_x} := \arg \max [Pr(\Delta_{\zeta_x} | \Xi, \gamma)]$

end

end

In Algorithm 1, T represents the input term list, O the ontology used in the semantic learning process, M the mapping from T to O and G a graph generator, and S_f represents the knowledge graph, which is constructed from the input term list. The stopping criterion for evaluating individual sets of rules can be any statistical measure of rule significance, such as for example the chi-squared metric, entropy-related measures or similar. The second while corresponds to a rule beam update, the key part of the semantic subgroup discovery.

There are two computationally expensive steps in the CBSSD approach. The community detection and the semantic subgroup discovery. The community detection algorithms used [21, 23] were previously proven to scale well up to millions of nodes and edges. The subgroup discovery part uses efficient beam search, where only a set of rules is propagated through search space and continuously upgraded. Furthermore, Hedwig [10, 12] uses efficient parallelism with bitsets.

As CBSSD consists of many distinct steps, having no free parameters would not cover all possible uses. First, the community detection step is parameterized in terms of number of iterations, as well as the detection type, which can include information on multilayer edges or not. Initial network construction is parameterized in the number of concurrent terms, being sent as a query to the BioMine graph crawler. Larger number of terms results in more coarse-grained networks and thus smaller numbers are preferred (e.g., in range from 1 to 10). Mind that smaller number of concurrent terms results in longer network construction step. Some of the key parameters for the rule learning part include beam size, search heuristic and significance thresholds. Larger beam sizes naturally result in larger rule-sets, which results in longer execution times.

3 Using CBSSD for Knowledge Discovery

This section demonstrates the use of the proposed methodology on two real world datasets from the life science domain. First, we consider properties of amino-acid variants within protein binding sites, followed by cancer related transcription factors identified in the context of epigenetics.

3.1 Discovery of Properties of Proteins with Single Amino-Acid Variants Present in the Binding Sites

Sequence variants are nucleotide or amino acid substitutions that can lead to unstable protein interaction complexes and thus influence the organism's phenotype (e.g., induce a disease state). There are two main types of variants: polymorphisms or germ-line variants that are heritable, and somatic mutations that appear in somatic tissues without previous genetic encoding. Although it was demonstrated that variants within biological interactions can be associated with disease occurrence [25–28], currently there are no studies of this phenomenon aimed at discovering new subgroups of proteins associated with variants within interaction sites at a more general level.

We use the results from a previous enrichment analysis study [25] for comparison with the proposed CBSSD methodology. Enrichment analysis in the context of this study is concerned with the identification of single significant terms, associated with the studied phenomenon. The results are compared based on terms appearing in both approaches, i.e. terms found as a result of enrichment analysis as well as a result of semantic subgroup discovery. As the two compared approaches are fundamentally different, the intersection of both results is expected to be relatively small (highly significant terms).

Preparing the Input for Semantic Subgroup Discovery. More than 300 UniProt terms, for which variants were found within protein binding sites, were used as the input query list (found in supplementary material of [25]). A BioMine knowledge graph with more than 1,650 nodes and 2,300 edges was constructed. The resulting network is depicted in Fig. 2.¹

Triplet construction consists of first mapping the nodes from the knowledge graph to the associated ontology terms, followed by the construction of the background knowledge. In this application, the Gene ontology [24] was used in both steps. Semantic subgroup discovery was conducted for more than 20 communities, and as the main result, more than 100 rules of various lengths were obtained. The most significant and the longest rules were manually inspected to identify possible overlap with previous pathway enrichment studies done on the same input dataset. Different beam sizes were experimented with in the procedure (from 10 to 50).

¹ Plotted with the Py3Plex library (<https://github.com/SkBlaz/Py3Plex>).

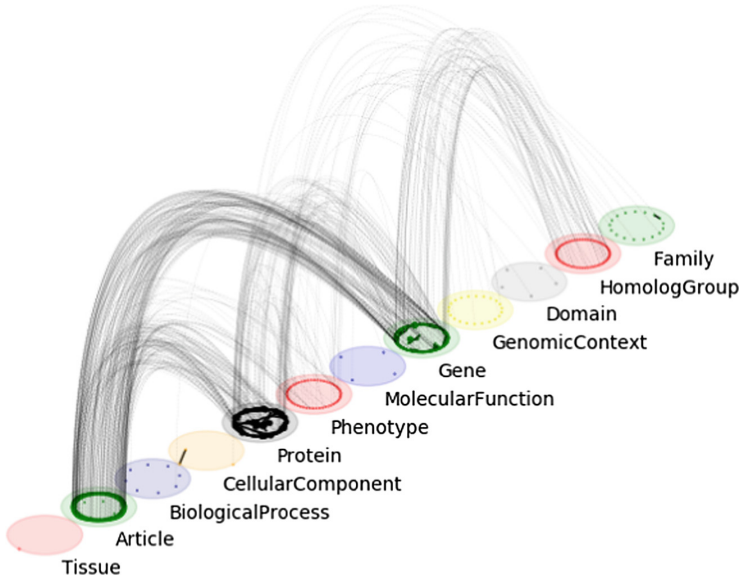


Fig. 2. Final size of the BioMine network, associated with polymorphisms located within protein interaction sites.

Results. The obtained rule sets for the identified communities were further inspected. We directly compared the ontology terms present in the rules with the terms, identified as significant in our previous study [25]. For this naïve comparison, conjuncts were considered as individual entries, as we were only interested in term presence (not coverage). There were 13 gene ontology terms present in both approaches (Table 1). Although only 13 terms were found with both procedures, the identified terms were among the most significant ones detected in the enrichment analysis setting. This indicates, that both procedures identified a strong signal related to DNA and cell cycle related processes. As semantic subgroup discovery was conducted for separate communities, the results were expected to be more detailed and comprehensive. This was indeed the case: given that many CBSSD rules consist of two conjuncts, these rules are potentially more informative than the ones identified by ontology enrichment analysis. As iron binding proteins were present in the protein list (this was known from the previous study [25]), rule $R = GO:0034618 \wedge GO:0006874$ appeared as one of the most significant rules ($p < 0.1$). Ontology terms in this rule represent arginine binding and cellular calcium homeostasis—both processes involving terms from the part of the input term list; a part not directly detected with enrichment analysis. The key UniProt term found for this rule was P41180 (CASR), which represents the extracellular calcium-sensing receptor [29]. As CASR is indeed critical for calcium homeostasis discovery (GO:0006874), it serves as an indicator of the validity of our CBSSD approach. The second term (GO:0034618), representing arginine binding is not

Table 1. Gene ontology terms, found both in enrichment and semantic rule learning process. Terms marked with * emerged as the most statistically significant ($p < 0.1$) and therefore relevant for semantic subgroup discovery.

Gene ontology term	Meaning
GO:0000077	DNA damage checkpoint*
GO:0000086	Mitotic cell cycle*
GO:0003677	DNA binding*
GO:0004871	Signal transducer activity*
GO:0005730	Nucleolus*
GO:0005814	Centriole
GO:0016020	Membrane
GO:0016605	PML body
GO:0030018	Z-disc
GO:0035264	Multicellular organism growth
GO:0045892	Negative regulation of transcription (DNA)
GO:0000122	Negative regulation of transcription (RNA)
GO:0000785	Chromatin

so directly associated with the CASR protein. To further investigate the context, within which GO:0034618 occurs, we queried the gene ontology database directly for similar proteins, already associated with this term. The majority of proteins, annotated with this term, correspond to acetylglutamate kinase, an enzyme that participates in the metabolism of amino acids (e.g., urea cycle). A possible interpretation of this association is that the CASR protein induces hormonal response, which could effectively lead to increased amino-acid metabolism, providing the molecular components necessary for establishment of homeostasis. This association serves as a possible candidate for further experimental testing and demonstrates the hypothesis generation capabilities of proposed approach.

Another interesting rule emerged from the first community identified. Rule $GO:0030903 \wedge GO:0000006$ was found for UniProt entries Q96SN8 (CDK5 regulatory subunit-associated protein 2), O94986 (Centrosomal protein), Q9HC77 (Centromere protein J) and O43303 (Centriolar coiled-coil protein). It can be observed that all the identified proteins are connected with nucleus-related processes. Term $GO:0030903$ corresponds to notochord development, which is a stage in cell division—a term directly associated with the identified proteins. The second term, $GO:0000006$, corresponds to high-affinity zinc uptake transmembrane transporter activity, a process related to enzyme system responsible for cell division and proliferation. Although this rule does not imply any new hypothesis, it demonstrates the generalization capability of the proposed approach.

Many terms remain specific for either semantic rule discovery based on community detection or enrichment analysis. This discrepancy appears due to the

fact that community detection splits the input term list into smaller lists, which can be described by completely different terms than the list as a whole. As the proposed methodology splits the input list, it is not sensible to compare it with conventional approaches, which operate on whole lists. Both approaches cover approximately the same percentage of input terms. The CBSSD's coverage is 12.02% with 218 GO terms, whereas the term coverage for conventional enrichment is 12.3% with 881 GO terms. The Term discrepancy serves only as a proof of fundamental difference between the two approaches. Nevertheless, we demonstrate that our approach is a useful complementary methodology to the well established enrichment analysis.

3.2 Grouping of Epigenetic Factors

Epigenetics is a field, where processes such as methylation are studied in the context of the influence of environment on the phenotype. Epigenetic factors are actively researched, and are constantly updated in databases such as emDB [30], where information such as gene expression, tissue information and variant information is publicly accessible. We tested the developed approach on the list of many currently known epigenetic factors related to cancer. The epigenetics dataset was chosen for two main reasons: first, to demonstrate the CBSSD's performance on a dataset, to our knowledge not yet used in semantic subgroup discovery, and second, this dataset serves to further test the developed methodology in the context of different biological process. The 153 distinct UniProt terms were used as input for the BioMine knowledge graph construction. Final graph consisted of approximately 4,500 nodes and 5,500 edges, respectively. The obtained knowledge graph is significantly larger than the one used in the previous case study (properties of SNVs in binding sites) and thus demonstrates the capabilities of the developed approach on larger graphs.

More than 50 communities were identified and further inspected. For the community including UniProt term Q8WTS6 (Histone-lysine N-methyltransferase), many interesting rules emerged. For example, rule ($p = 0.09$): $GO:1990785 \wedge GO:0000975 \wedge GO:0000082$ indicates that the protein is indeed highly associated with epigenetic processes. Term $GO:1990785$ describes water-immersion restraint stress, term $GO:0000975$ regulatory region DNA binding and term $GO:0000082$ transition of mitotic cell cycle. All three terms describe the Q8WTS6 entry, as it effects the DNA's topological properties (coil formation) and is responsible for transcriptional activation of genes, which code for collagenases, enzymes crucial to mitotic cell cycle (wall formation). To further analyze CBSSD's generalization capabilities, we plotted all rule sets (communities) against all GO terms, identified as enriched by the DAVID Bioinformatics Suite [31]. As this experiment is conducted using only terms, previously identified as significant, CBSSD's significance threshold was relaxed to $p = 0.5$. Additional relaxation was introduced to cover more possibly interesting patterns, which would otherwise be considered noise or false positive results.

The semantic landscape obtained in this experiment is depicted in Fig. 3. It can be observed that only a handful of GO terms serve as a basis for more

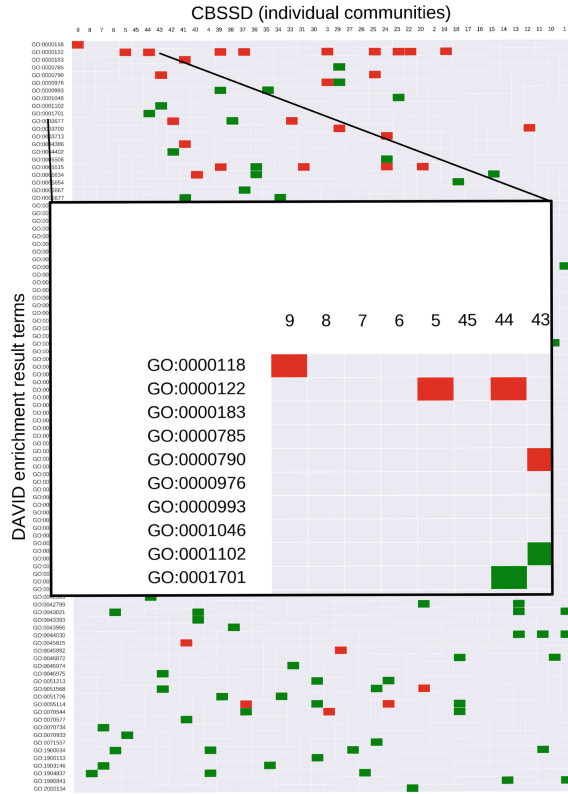


Fig. 3. Visualizing higher order abstraction emerging from previously enriched terms associated with epigenetic regulators. It can be observed (inset image), that only a couple of terms correspond to multi-term rules (red rectangles). Terms, such as *GO:0000118* represent very high level terms, associated with majority of epigenetics-related processes. Such terms are most commonly included in more complex rules. (Color figure online)

complex rules. For this example, some of these terms are *GO:0000118*, which represents the Hystone deacetylase complex, one of the key mechanisms for hystone structure regulation. The *GO:0000112*, representing negative regulation of transcription from RNA polymerase II promoter, a mechanism by which many epigenetic regulators influence the transcription patterns, *GO:0000183*, representing chromatin silencing at rDNA, *GO:0000785* and *GO:0000790*, representing chromatin in general, *GO:0000976*, representing transcription regulatory region sequence-specific DNA binding and *GO:0001046*, which represents core promoter sequence-specific DNA binding. The described terms are all fundamentally associated with epigenetic regulation, which proves CBSSD was able to use the more general terms to construct meaningful rules. Overall, 27% of all significant terms identified via conventional enrichment analysis were also

found via CBSSD algorithm. Such low percentage is expected, as CBSSD builds upon individual subsets of the larger termset, used in conventional enrichment. This result implies the higher level terms are similar in both approaches, yet CBSSD identified latent patterns, which can not be detected via conventional enrichment. The higher level terms appear to form the base for more complex rules. Similar behavior was reported as a result of the SegMine methodology [32], which similarly to CBSSD yields explanatory power of rules in order to find enriched parts of input term lists. Coverage-wise, both approaches perform the same, as the CBSSD's coverage is 96.7% with 230 GO terms, whereas the term coverage for conventional enrichment is 96.7% with 360 GO terms. Similarly to the case study one, CBSSD needed less GO terms to cover approximately the same percentage of input term list.

4 Conclusions and Further Work

Semantic data mining is an emerging field, where background knowledge in the form of ontologies can be used to generalize the rules emerging from the learning process. In this study, we demonstrate how such an approach can be used to induce rules describing the communities, detected on an automatically constructed knowledge graph. Our implementation was tested on two data sets from the life science domain, where validity of the most significant rules was manually inspected in terms of biological context. This approach works for up to 6,000 terms in reasonable time (e.g., a day), but for more than e.g., 10,000 terms, whole graphs should be used from the beginning, if possible. As the number of rules produced can be large, adequate visualization techniques for elegant result inspection are still to be developed. Our approach differs significantly from conventional enrichment analysis, as interesting groups of terms are identified based on the underlying network structure, rather than manual, expert-guided selection. We currently see CBSSD it as a complementary methodology to enrichment analysis, as it is capable of describing latent patterns beyond the ones expected by a domain expert. Further work includes extensive testing of CBSSD on larger data sets, possibly from many different domains.

Availability. The Community-based subgroup discovery reference implementation is freely available at <https://github.com/SkBlaz/CBSSD>.

Acknowledgments. This research was funded by the Slovenian Research Agency funded project *HinLife: Analysis of Heterogeneous Information Networks for Knowledge Discovery in Life Sciences* (J7-7303), as well as the *The Human Brain Project* (FET Flagship grant FP7-ICT-604102). The authors also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan-XP GPU used for this research.

References

1. Drummond, A.J., Rambaut, A.: Beast: Bayesian evolutionary analysis by sampling trees. *BMC Evol. Biol.* **7**(1), 214 (2007)
2. Madahian, B., Deng, L., Homayouni, R.: Development of a literature informed Bayesian machine learning method for feature extraction and classification. *BMC Bioinform.* **16**(Suppl. 15), P9 (2015)
3. Lavrač, N., Džeroski, S.: *Inductive Logic Programming* (1994)
4. Vavpetič, A., Lavrač, N.: Semantic subgroup discovery systems and workflows in the SDM-toolkit. *Comput. J.* **56**(3), 304–320 (2012)
5. Balcan, N., Blum, A., Mansour, Y.: Exploiting structures and unlabeled data for learning. In: *Proceedings of the 30th International Conference on International Conference on Machine Learning, ICML 2013*, vol. 28, pp. 1112–1120 (2013)
6. Liu, H., Dou, D., Jin, R., LePendou, P., Shah, N.: Mining biomedical ontologies and data using RDF hypergraphs. In: *2013 Proceedings of the 12th International Conference on Machine Learning and Applications (ICMLA)*, vol. 1, pp. 141–146. IEEE (2013)
7. Hotho, A., Staab, S., Stumme, G.: Ontologies improve text document clustering. In: *Third IEEE International Conference on Data Mining*, pp. 2–5 (2003)
8. Belleau, F., Nolin, M.A., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: towards a mashup to build bioinformatics knowledge systems. *J. Biomed. Inf.* **41**(5), 706–716 (2008)
9. Eronen, L., Toivonen, H.: Biomine: predicting links between biological entities using network models of heterogeneous databases. *BMC Bioinform.* **13**(1), 119 (2012)
10. Vavpetič, A., Novak, P.K., Grčar, M., Mozetič, I., Lavrač, N.: Semantic data mining of financial news articles. In: Fürnkranz, J., Hüllermeier, E., Higuchi, T. (eds.) *DS 2013. LNCS (LNAI)*, vol. 8140, pp. 294–307. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40897-7_20
11. Langohr, L., Podpečan, V., Petek, M., Mozetič, I., Gruden, K., Lavrač, N., Toivonen, H.: Contrasting subgroup discovery. *Comput. J.* **56**(3), 289–303 (2012)
12. Adhikari, P.R., Vavpetič, A., Kralj, J., Lavrač, N., Hollmén, J.: Explaining mixture models through semantic pattern mining and banded matrix visualization. *Mach. Learn.* **105**(1), 3–39 (2016)
13. Cohen, R., Havlin, S.: *Complex Networks: Structure, Robustness and Function*. Cambridge University Press, Cambridge (2010)
14. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. *arXiv preprint physics/0506133* (2005)
15. Vrabčič Rok, H.D., Butala, P.: Discovering autonomous structures within complex networks of work systems. *CIRP Ann. Manuf. Technol.* **61**(1), 423–426 (2012)
16. Strogatz, S.H.: Exploring complex networks. *Nature* **410**(6825), 268 (2001)
17. Duch, J., Arenas, A.: Community detection in complex networks using extremal optimization. *Phys. Rev. E* **72**(2), 027104 (2005)
18. The UniProt Consortium, et al.: UniProt: the universal protein knowledgebase. *Nucleic Acids Res.* **45**(D1), D158–D169 (2017)
19. Kanehisa, M., Goto, S.: Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.* **28**(1), 27–30 (2000)
20. Benson, D.A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Sayers, E.W.: Genbank. *Nucleic Acids Res.* **41**(D1), D36–D42 (2012)

21. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**(10), P10008 (2008)
22. Newman, M.E.: Modularity and community structure in networks. *Proc. Nat. Acad. Sci.* **103**(23), 8577–8582 (2006)
23. Rosvall, M., Axelsson, D., Bergstrom, C.T.: The map equation. *Eur. Phys. J. Spec. Topics* **178**(1), 13–23 (2009)
24. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., et al.: Gene ontology: tool for the unification of biology. *Nat. Genet.* **25**(1), 25–29 (2000)
25. Škrlić, B., Konc, J., Kunej, T.: Identification of sequence variants within experimentally validated protein interaction sites provides new insights into molecular mechanisms of disease development. *Mol. Inform.* **36**, 1–8 (2017)
26. Škrlić, B., Kunej, T.: Computational identification of non-synonymous polymorphisms within regions corresponding to protein interaction sites. *Comput. Biol. Med.* **79**, 30–35 (2016)
27. Schröder, N.W., Schumann, R.R.: Single nucleotide polymorphisms of toll-like receptors and susceptibility to infectious disease. *Lancet Infect. Dis.* **5**(3), 156–164 (2005)
28. Kamburov, A., Lawrence, M.S., Polak, P., Leshchiner, I., Lage, K., Golub, T.R., Lander, E.S., Getz, G.: Comprehensive assessment of cancer missense mutation clustering in protein structures. *Proc. Nat. Acad. Sci.* **112**(40), E5486–E5495 (2015)
29. Garrett, J.E., Capuano, I.V., Hammerland, L.G., Hung, B.C., Brown, E.M., Hebert, S.C., Nemeth, E.F., Fuller, F.: Molecular cloning and functional expression of human parathyroid calcium receptor cDNAs. *J. Biol. Chem.* **270**(21), 12919–12925 (1995)
30. Nanda, J.S., Kumar, R., Raghava, G.P.: dbEM: a database of epigenetic modifiers curated from cancerous and normal genomes. *Sci. Rep.* **6**, 19340 (2016)
31. Huang, D.W., Sherman, B.T., Tan, Q., Kir, J., Liu, D., Bryant, D., Guo, Y., Stephens, R., Baseler, M.W., Lane, H.C., et al.: David bioinformatics resources: expanded annotation database and novel algorithms to better extract biology from large gene lists. *Nucleic Acids Res.* **35**(2), W169–W175 (2007)
32. Podpečan, V., Lavrač, N., Mozetič, I., Novak, P.K., Trajkovski, I., Langohr, L., Kulovesi, K., Toivonen, H., Petek, M., Motaln, H., et al.: Segmine workflows for semantic microarray data analysis in Orange4WS. *BMC Bioinform.* **12**(1), 416 (2011)

Author Index

- Agapito, Giuseppe 1
Arbajian, Pierre 16
- Barracchia, Emanuele Pio 35
Basile, Teresa M. A. 49
Biondo, Daniele 107
Boubekki, Ahcène 167
Branders, Vincent 65
Brbić, Maria 138
Brefeld, Ulf 167
- Cannataro, Mario 1
Ceci, Michelangelo 35
- Di Mauro, Nicola 49
Dînşoreanu, Mihaela 80, 93
Dolean, Samuel 80
Dupont, Pierre 65
Džeroski, Sašo 138, 151
- Esposito, Floriana 49
- Ferilli, Stefano 49
- Geiszt, Attila 80
Graur, Dan-Ovidiu 93
Guarascio, Massimo 107
Guzzi, Pietro H. 1
- Hajja, Ayman 16
- Kocev, Dragi 138, 151
Kralj, Jan 182
- Laurinec, Peter 122
Lavrač, Nada 182
- Lemnaru, Camelia 93
Levatić, Jurica 138
Lucká, Mária 122
- Malerba, Donato 35
Mammoliti, Rocco 107
Mariş, Răzvan-Alexandru 93
Mureşan, Raul Cristian 80
- Nikoloski, Stevanche 151
- Perdih, Tomaž Stepišnik 138
Pio, Gianvito 35
Potolea, Rodica 80, 93
- Raś, Zbigniew W. 16
Reubold, Jan 167
Ritacco, Ettore 107
- Schaus, Pierre 65
Škrlj, Blaž 182
Šmuc, Tomislav 138
Strufe, Thorsten 167
Supek, Fran 138
- Țincaş, Ioana 80
Toma, Alessandra 107
- Vavpetič, Anže 182
Vergari, Antonio 49
Vidulin, Vedrana 138
- Wieczorkowska, Alicja A. 16