Francesco Masulli · Alfredo Petrosino
Stefano Rovetta (Eds.)

LNCS 7627

Clustering High-Dimensional Data

First International Workshop, CHDD 2012
Naples, Italy, May 15, 2012
Revised Selected Papers

Springer

Lecture Notes in Computer Science 7627

Francesco Masulli · Alfredo Petrosino
Stefano Rovetta (Eds.)

Clustering High-Dimensional Data

First International Workshop, CHDD 2012
Naples, Italy, May 15, 2012
Revised Selected Papers

Springer

Editors

Francesco Masulli
DIBRIS
University of Genoa
Genoa
Italy

Stefano Rovetta
DIBRIS
University of Genoa
Genoa
Italy

Alfredo Petrosino
University of Naples "Parthenope"
Naples
Italy

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-662-48576-7 ISBN 978-3-662-48577-4 (eBook)
DOI 10.1007/978-3-662-48577-4

Library of Congress Control Number: 2015950900

LNCS Sublibrary: SL3 – Information Systems and Applications, incl. Internet/Web, and HCI

Springer Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer-Verlag GmbH Berlin Heidelberg is part of Springer Science+Business Media
(www.springer.com)

Preface

One of the most long-standing problems afflicting machine learning techniques is dataset dimensionality. Owing to the evolution of technologies for acquiring and creating information, however, this issue has recently become ubiquitous. In many applications to real-world problems, we deal with data with anywhere from a few dozen to many thousands of dimensions. Such high-dimensional data spaces are often encountered in areas such as medicine or biology, where DNA microarray technology and next-generation sequencing can produce a large number of measurements at once; the clustering of text documents, where, if a word-frequency vector is used, the number of dimensions equals the size of the dictionary; and many others, including data integration and management, and social network analysis. In all these cases, the dimensionality of data makes learning problems hardly tractable.

In particular, dimensionality is a highly critical factor for the clustering task. The following problems need to be addressed for clustering high-dimensional data:

– When the dimensionality is high, the volume of the space increases so fast that the available data become sparse, and we cannot find reliable clusters, as clusters are data aggregations (curse of dimensionality).
– The concept of distance becomes less precise as the number of dimensions grows, since the distance between any two points in a given dataset converges (concentration effects).
– Different clusters might be found in different subspaces, thus a global filtering of attributes is not sufficient (local feature relevance problem).
– Given a large number of attributes, it is likely that some attributes are correlated. Hence, clusters might exist in arbitrarily oriented affine subspaces.
– High-dimensional data could likely include irrelevant features, which may obscure the effect of the relevant ones.

This volume is the outcome of work done during the International Workshop on Clustering High-Dimensional Data, held at Istituto Italiano per gli Studi Filosofici, Palazzo Serra di Cassano, in Naples (Italy) on May 15, 2012, where speakers were subsequently invited to submit a paper related to their presentation.

The papers collected here aim to present an updated view of many different approaches toward clustering high-dimensional data, and can be divided by topic into three groups.

The first group introduces the general subject and issues of high-dimensional data clustering. Chapter 1 provides a general introduction, while Chapter 2 explores some properties of high-dimensional data that make it difficult to detect and even to define clusters.

The second group of chapters presents examples of techniques used to find and investigate clusters in high dimensionality. Chapter 3 focuses on an approach to *subspace clustering*; Chapter 4 presents a selection of dimensionality-independent methods for comparing clusterings; and Chapter 5 deals with clustering high-dimensional time series.

The third group deals with the most common approach to tackling dimensionality problems, namely, dimensionality reduction and its application in clustering. Chapter 6 introduces the topic of *intrinsic dimensionality estimation*, and Chapter 7 presents a specific technique for intrinsic dimensionality estimation. Chapter 8 compares four dimensionality reduction methods for binary data, while the last contribution, Chapter 9, focuses on dimensionality reduction by feature selection using rough-fuzzy techniques.

July 2015

Francesco Masulli
Alfredo Petrosino
Stefano Rovetta

Organization

The International Workshop on Clustering High-Dimensional Data was organized as part of the Project “Clustering di dati ad alta dimensionalità” funded by the GNCS - Istituto Nazionale di Alta Matematica Francesco Severi (IN-dAM), in collaboration with the Istituto Italiano per gli Studi Filosofici (Naples, Italy), the Special Interest Group in Bioinformatics and Intelligence of INNS, the Task Force on Neural Networks of IEEE-CIS-TCBB, the Department of Computer and Information Sciences of the University of Genoa (Italy), and the Department of Applied Science, University of Naples Parthenope (Italy).

Workshop Chairs

Francesco Masulli	University of Genoa, Italy and Temple University, Philadelphia, USA
Alfredo Petrosino	University of Naples Parthenope, Italy

Scientific Secretary

Stefano Rovetta	University of Genoa, Italy

Event Management

Hassan Mahmoud	University of Genoa, Italy

This workshop has been made possible thanks to funding from GNCS - Gruppo Nazionale per il Calcolo Scientifico of the INdAM - Istituto Nazionale di Alta Matematica Francesco Severi.

We warmly thank the Istituto Italiano di Studi Filosofici for supporting the event.

Contents

Clustering High-Dimensional Data

Francesco Masulli[1,2](✉) and Stefano Rovetta[1]

[1] Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi DIBRIS, Università di Genova, Genova, Italy
[2] Center for Biotechnology, Temple University, Philadelphia, USA
francesco.masulli@unige.it

Abstract. This chapter introduces the task of clustering, concerning the definition of a structure aggregating the data, and the challenges related to its application to the unsupervised analysis of high-dimensional data. In the recent literature, many approaches have been proposed for facing this problem, as the development of efficient clustering methods for high-dimensional data is is a great challenge for Machine Learning as it is of vital importance to obtain safer decision-making processes and better decisions from the nowadays available Big Data, that can mean greater operational efficiency, cost reduction and risk reduction.

1 Introduction

Clustering aims to find a structure that aggregates the data into some groups with the property that data belonging to a group (or cluster) are more similar to data in that cluster than to data in other clusters.

With the beginning of the 21st Century, the decrease in the cost of storage and the increasing of the interest that is permeating the society toward the collection of data of all kinds, on scales unimaginable until recently, in most of the fields, ranging from science, to finance, to the Internet and mobile devices and sensors, has resulted in the availability of large, continuously growing masses of data (Big Data).

Often, those data contain hyper-informative details about each observed instance. The problem of data clustering in high-dimensional data spaces has then become of vital interest for the analysis of those Big Data, to obtain safer decision-making processes and better decisions.

This chapter is organized as follows: Sect. 2 introduces the problem of clustering; Sect. 3 presents the problem of high-dimensional data analysis; Some relevant approaches high-dimensional data clustering are surveyed in Sect. 4; Sect. 5 presents the conclusions.

2 Defining Clustering

The concept of *clustering* dates back to at least the Greeks philosophers. Plato (∼400 BC), in his *Statesman* dialogue [15], introduces the approach of grouping objects based on their similar properties (categorization). This approach was

F. Masulli et al. (Eds.): CHDD 2012, LNCS 7627, pp. 1–13, 2015.
DOI: 10.1007/978-3-662-48577-4_1

further explored and systematized by Aristotle (∼350 BC) in his *Categories* treatise [6].

In the last century, psychologists of Gestalt (word that in German means “shape, form”) proposed a theory of visual perception organization based on the so called *Principles of Grouping* [25,26,42]:

– *Law of Proximity*: perception tends to group stimuli that are close together as part of the same object, and stimuli that are far apart as two separate objects.

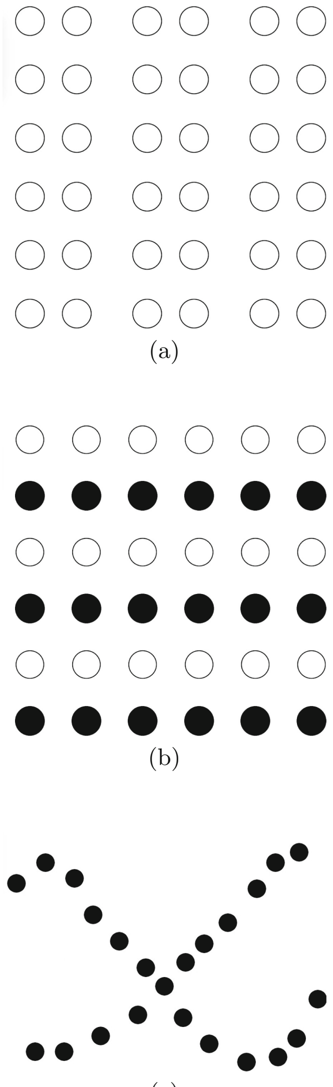


Fig. 1. Gestalt’s Laws: (a) Law of Proximity; (b) Law of Similarity; (c) Law of Good Continuation.

– *Law of Similarity*: perception lends itself to seeing stimuli that physically resemble each other as part of the same object, and stimuli that are different as part of a different object.
– *Law of Good Continuation*: when there is an intersection between two or more objects, people tend to perceive each object as a single uninterrupted object.
– and other laws (*Closure*, *Good Form*, *Common Fate*, etc.).

As shown in the examples in Fig. 1 those Gestalt's law explain many cases when our perceptual system organizes the elements of a visual stimulus in separate groups (or *clusters*).

From a Machine Learning viewpoint, an intuitive definition of *clustering* task can be: *To find a structure in the given data that aggregates the data into some groups (or clusters) with the property that data belonging to a cluster are more similar to data in that cluster than to data in other clusters* (homogeneity criterion). A more formal definition of clustering is difficult to state, as the clustering task is not a well-posed problem [23,38].

A problem is well-posed in Hadamard's sense [21] when a solution exists, is unique, and depends continuously on the initial data (i.e., is robust against noise). Form this point of view, clustering is a ill-posed problems, as the partitioning task of data sets following the homogeneity criterion can lead to different reasonable solutions, and, in addition, the solution obtained by many clustering algorithms are subject to the noise (outliers) in the data. Figure 2 illustrates a typical situation in clustering: We have a two-dimensional data set of 20 points (a) that can be easily portioned in two groups (b), but, if we allow clusters to be nested, we can interpret it as a structure of six clusters (c), or also as a different organization of four clusters (d).

Many measures of cluster validity have been introduced in the literature of the last four decades, proposing endogenous and exogenonous criteria [29] for choosing the best clustering, but often the best clustering depends on the desired results and we embed our knowledge on the problem (implicitly or explicitly) in the clustering algorithm, performing a regularization of the problem [21].

To manage the complexity and ambiguity of the problem we could start with an operational definition of a clustering procedure. To this aim, given the data set, we state a clustering problem by setting the following elements:

– A *data representation*: we must represent the instances of the data set as vectors of characteristics (or features) using a suitable transformation of raw data. We must choose the feature to use and their attributes (binary, discrete, continue) and scales. Data representation is usually strictly dependent on the knowledge of the problem. Sometimes the features are given, while sometimes we need to choose/select them. We stress that a good choice of features can lead to a good quality of clustering procedure.
– A *similarity measure* (or, conversely, a dissimilarity measure) to be used for comparing the instances of the data set. Among the most used measures, we can mention the Euclidean, the Mahalanobis, and the Hamming distances (that are dissimilarity measures), and similarity measures such as the correlation and the Jacard index [23].

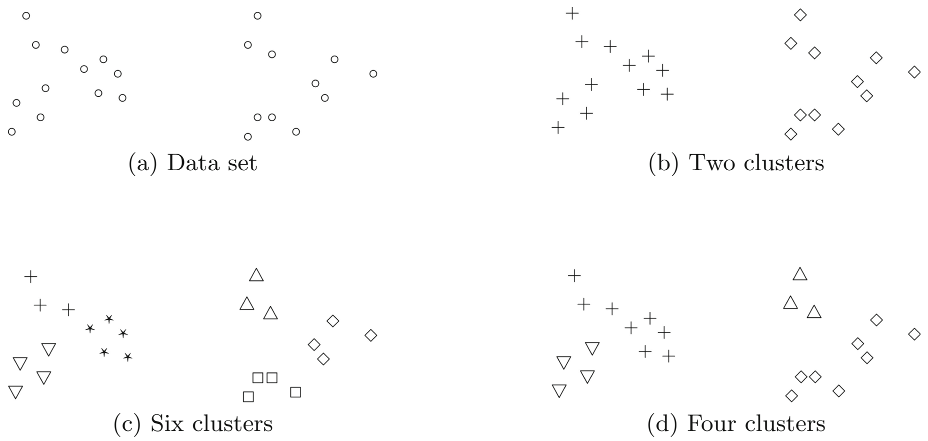


Fig. 2. Different clusterings for a set of points (modified from [38]).

– A *model of clustering*, i.e., a clustering algorithm. The principal categories of clustering algorithms are the *hierarchical algorithms* that try to find structures of data which can be further divided in substructures recursively, and *partitive algorithms* that are aimed to find a single partition of data. In the choice of the clustering model we should also take into account that most clustering algorithms, such as the K-Means, seek to find a classification of the data into non-overlapping groups [29]; conversely, fuzzy clustering algorithms allow an object to partially belong to several groups [8].

This is our reference setting in the remainder of the chapter.

We apply clustering techniques in data analysis instead of supervised classification techniques (e.g. neural networks or support vector machines) [23] when labeling the (full) data set is too expensive or infeasible, when the available labeling of data is ambiguous, when we need to improve our understanding of the nature of data, or we want to reduce the number of data (information) to transmit. For those reasons clustering is a fundamental tool in data mining, document retrieval, image segmentation and pattern classification.

3 The Century of Big Data

In the last decades the quantity of data to be analyzed has increased explosively due to socials factors and to the related technological evolution (CPU, memory storage systems, computer interconnections) which showed a growth sometime faster than that exponential expected from Moore's Law [34].

In this regard we quote here some passages from the speech given by David L. Donoho at the *AMS Conference on Math Challenges of the 21st Century* in 2000 [18]: "*The coming century is surely the century of data. A combination of blind faith and serious purpose makes our society invest massively in the collection and processing of data of all kinds, on scales unimaginable until recently.*

Hyperspectral Imagery, Internet Portals, Financial tick-by-tick data, and DNA Microarrays are just a few of the better-known sources, feeding data in torrential streams into scientific and business databases worldwide. ...

The trend today is towards more observations but even more so, to radically larger numbers of variables – voracious, automatic, systematic collection of hyper-informative detail about each observed instance. We are seeing examples where the observations gathered on individual instances are curves, or spectra, or images, or even movies, so that a single observation has dimensions in the thousands or billions, while there are only tens or hundreds of instances available for study. ..."

One year later, in 2001, Douglas Laney of Gartner, Inc. proposed the now mainstream definition of *Big Data* as the three Vs of Big Data (*"3Vs"* model): *volume* (amount of data), *velocity* (speed of data in and out) and *variety* (range of data types and sources) [30]. In addition, some other dimensions are considered when thinking about Big Data, included *variability* (as data flows can be highly inconsistent with periodic peaks), *veracity* (as the quality of the data being captured can vary greatly), and *complexity* (as data coming from multiple sources need to be linked, connected and correlated before analysis).

Big Data is nowadays a popular term used to describe the exponential growth and availability of data, both structured and unstructured. Big Data may be as important to business and society as the Internet has become, as more data may lead to more accurate analyses, and more accurate analyses may lead to more confident decision making and better decisions can mean greater operational efficiencies, cost reductions and reduced risk [31].

While in the past excessive data volume was a storage issue, with the present fast decreasing trend of storage costs other issues emerge. As Donoho pointed out in [18], "*classical methods are simply not designed to cope with this kind of explosive growth of dimensionality of the observation vector. We can say with complete confidence that in the coming century, high-dimensional data analysis will be a very significant activity, and completely new methods of high-dimensional data analysis will be developed; we just don't know what they are yet. ...*"

Concerning high-dimensional data analysis, we quote an excerpt from page 97 of the well-known book *Adaptive Control Processes: A Guided Tour* by Richard Bellman, published in 1961 [7], where he introduces the concept of *curse of dimensionality* as the impossibility of optimizing a function of many variables by a brute force search on a discrete multidimensional grid, as the number of grids points increases exponentially with dimensionality, i.e., with the number of variables: "*In view of all that we have said in the forgoing sections, the many obstacles we appear to have surmounted, what casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientist from the earliest days*".

Nowadays, the *curse of dimensionality* refers to any problem in data analysis that results from a large number of variables (features, or attributes).

When we cluster high dimensional data sets we have to overcome some difficult problems that are expressions of the *curse of dimensionality* [27]:

- When the dimensionality is high, the volume of the space increases so fast that the available data becomes sparse, and we cannot find reliable clusters, as clusters are data aggregations.
- The concept of distance becomes less precise as the number of dimensions grows, since the distance between any two points in a given dataset converges (*concentration effect*):

$$\lim_{d \to +\infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \to 0$$

- Different clusters might be found in different subspaces, so a global filtering of attributes is not sufficient (*local feature relevance problem*).
- Given a large number of attributes, it is likely that some attributes are correlated. Hence, clusters might exist in arbitrarily rotated subspaces.
- High-dimensional data could likely include irrelevant features, which may obscure the effect of the relevant ones.

4 Approaches to High Dimensional Data Clustering

4.1 Subspace Clustering

Subspace clustering algorithms localize the search for relevant dimensions allowing them to find clusters that exist in multiple, and possibly overlapping subspaces. This technique is an extension of feature selection that attempts to find clusters in different subspaces of the same dataset. Just as with feature selection, subspace clustering requires a search method and an evaluation criteria. In addition, subspace clustering must somehow limit the scope of the evaluation criteria so as to consider different subspaces for each different cluster.

There are two major approaches to subspace clustering based on their search strategy:

- Top-down algorithms that find an initial clustering in the full set of dimensions and evaluate the subspaces of each cluster, and then iteratively improve the results.
- Bottom-up approaches that find dense regions in low dimensional spaces and combine them to form clusters.

Subspace clustering is aimed to find all clusters in all subspaces. In different subspaces the same point can then belong to different clusters. Subspaces can be axis-parallel or general (i.e., not necessarily axis-parallel). In a space with d dimensions we have $2^d - 2$ axis-parallel subspaces, not counting the space itself and the empty (0-dimensional) degenerate subspace. If we allow rotations in addition to axis selection, then an infinite number of subspaces is possible. Therefore to make Subspace Clustering computationally feasible some heuristic must be used, such as in CLIQUE [5] and SUBCLU [24].

4.2 Projected Clustering

In high-dimensional spaces, even though a good partition cannot be defined on all the dimensions because of the sparsity of the data, some subset of the dimensions can always be obtained on which some subsets of data form high quality and significant clusters.

Projected clustering [2,4] methods are aimed to find clusters (referred to as projected clusters) specific to a particular group of dimensions. Each cluster may refer to a different subsets of dimensions

The output of a typical projected clustering algorithm, searching for k clusters in subspaces of dimension l, is twofold:

– a partition of data of $k+1$ different clusters, where the first k clusters are well shaped, while the $(k+1)$-th cluster elements are outliers, which by definition do not cluster well.
– a possibly different set of l dimensions for each of the first k clusters, such that the points in each of those cluster are well clustered in the subspaces defined by these vectors.

Usually, projected clustering algorithms use a special distance function together with a standard clustering algorithm. For example, the PreDeCon algorithm [10] after checking the attributes that best support a cluster for each point, applies a standard clustering algorithm, using a modified distance function weighting more the dimensions with low variance. Another projected clustering algorithm, named PROCLUS [2], proceeds as the regular Partitioning Around Medoids (PAM) algorithm, but for each found medoid it finds the subspace spanned by attributes with low variance, and then points are assigned to the closest medoid, considering only the subspace of that medoid in determining the distance.

4.3 Biclustering

Biclustering [22,32] (aka co-clustering, or two-way clustering) is a methodology allowing for feature set and data points clustering simultaneously, i.e., to find clusters of samples possessing similar characteristics together with features creating these similarities.

In other words, biclustering answers the question: What characteristics make similar objects similar to each other? The output of biclustering is not a partition or hierarchy of partitions of either rows or columns, but a partition of the whole matrix into sub-matrices or patches. We can obtain different biclustering structures: single bicluster, different non-overlapping structures, and overlapping with or without structure.

The goal of biclustering is to find as many patches as possible, and to have them as large as possible, while maintaining strong homogeneity within patches.

In the microarray analysis framework, the pioneering work by Cheng and Church [13] employs a set of greedy algorithms to find one or more biclusters in gene expression data, based on a mean squared residue as a measure of similarity.

Table 1. Shared farthest neighbor clustering algorithm.

1. Compute the distance (or other dis/similarity measure) of each point to all others, and for each point identify the farthest neighbor (labeling).
2. Find the clusters at the first level by aggregating all points sharing a common farthest neighbor label.
3. Within each cluster, the second farthest neighbor can be considered in the same way. This produces a second level clustering within each cluster of the first level.
4. The procedure is recursively repeated until no further differentiation is found (all points within a level l-1 cluster share the same l-th farthest neighbor), or until a predefined maximum level is reached.

One bicluster is identified at a time iteratively. The masking of null values of the discovered biclusters are replaced by large random numbers that help to find new biclusters at each iteration. Nodes are deleted and added and also the inclusion of inverted data is taken into consideration when finding biclusters. [40] notices that this approach affects the subsequent discovery of large-sized biclusters and proposes a two-phase probabilistic algorithm termed Flexible Overlapped Clusters (FLOC) to simultaneously discover a set of possibly overlapping biclusters. Bipartite graphs are also employed in [39], with a bicluster being defined as a subset of genes that jointly respond across a subset of conditions. An approach based on fuzzy sets theory is presented in [20]. Other methods for biclustering have been also presented in [12,33,43] and in many more recent papers.

4.4 Hierarchical Clustering

Hierarchical clustering (also called hierarchical cluster analysis or HCA) methods which seek to organize data in a hierarchy of clusters. There are two main strategies for hierarchical clustering [35]:

- Agglomerative or "bottom up" approaches, where each instance starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- Divisive or "top down" approaches, where all instances start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In general, the merges and splits are determined in a greedy manner. Hierarchical clustering a shows the capability of visualizing a structure in the form of a taxonomy (dendrogram), which makes it interesting in a number of applications.

Efficient agglomerative methods of complexity $O(n^2)$ are SLINK [37] for single-linkage and CLINK [17] for complete-linkage clustering. Eisen et al. [19] proposed a hierarchical agglomerative clustering algorithm whereby at each stage only two objects (either single data objects or clusters) are merged. The algorithm operates on a proximity matrix containing the distances among all possible pairs of points.

Hierarchical approach to clustering work is useful for high dimensional and low cardinality data sets, but the taxonomy obtained is not very stable, as often a

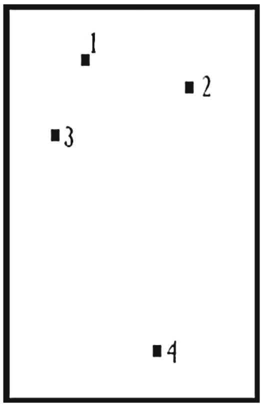


Fig. 3. Data points.

small noise on data can significantly affect it. Moreover, the standard hierarchical approach does not require the selection of model order, simply because it makes no attempt at defining "real" clusters (densities) and provides an organization of data, but with no identification of clusters. Finally, agglomerative algorithms cannot produce a partial (rough) result, to be refined only if needed ("anytime" algorithms in the data mining jargon).

Table 2. (a) Data matrix; (b) Proximity matrix; (c) Rank matrix.

Data point	x	y
1	2	8.3
2	4.7	7.4
3	1.0	6.3
4	3.6	1.2

(a)

Data point	1	2	3	4
1	0	2.85	2.32	7.28
2	2.85	0	3.89	6.30
3	2.32	3.89	0	5.64
4	7.28	6.30	5.64	0

(b)

Data point	1	2	3	4
I Neighboor	3	1	1	3
II Neighboor	2	3	2	2
III Neighboor	4	4	4	1

(c)

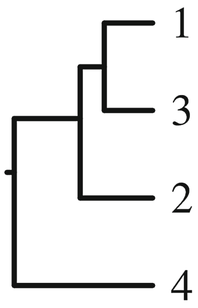


Fig. 4. Dendrogram for the problem shown in Fig. 3.

In [36] we proposed a different approach to hierarchical clustering based on the so-called *Principle of Points in Perspective* (3P): *Two points should be considered similar if they share the same farthest point among all remaining data*, i.e., points must be examined not with reference to their neighborhood (locally), but with reference far-away points in the data set, therefore *"in perspective"*. Then, we iterate to the second far way, the third, and so on. For each point we will use the list of ranks of other points.

The method for hierarchical clustering we proposed in [36], named *Shared Farthest Neighbor* (SFN) clustering algorithm, is based on the following property of data: Closer points have usually a different nearest neighbor and equal farthest neighbor. In Table 1 a description of the SFN clustering algorithm is presented.

In Fig. 3 and Table 2 we show an example. The coordinates of data points are reported in Table 2a. We calculate the Proximity matrix (Table 2b) using the Euclidean distance, and then the Rank matrix (Table 2c). In Fig. 4 the dendrogram resulting from the application of SFN clustering algorithm to the Rank matrix is shown.

We notice that the Shared Farthest Neighbor approach to clustering gives us a stable dendrogram result, as it start working with the farthest points, and that evaluation is less sensible to noise as the closeness evaluation used in standard hierarchical clustering methods. Moreover, we can stop the iteration of the SFN algorithm "anytime", obtaining a partial analysis of the data points aggregation.

In [36] we evaluated the quality of the SFN clustering algorithm on a group of biomedical data sets, choosing among labeled benchmarks and using labels for external evaluation. The obtained error ratios of SFN are comparable with or lower than those obtained in literature using supervised techniques.

5 Conclusions

In this chapter we have introduced the problem of data clustering in high-dimensional data space that in recent years has become of vital interest for the analysis of so called Big Data, term describing growth and availability of data in most of the fields, ranging from science, to finance, to Internet and sensors and mobile devices, both structured and unstructured.

We provided also a quick suvery of some approaches to High Dimensional Data Clustering, including Subspace Clustering, Projected Clustering, Biclustering, and Hierarchical Clustering, giving some details on *Shared Farthest Neighbor* algorithm for agglomerative hierarchical clustering proposed by us [36]. Many other approaches towards clustering high-dimensional data presented in the recent years are noteworthy, as, e.g., the techniques of correlation clustering [1,3,11], ensemble clustering [28], and multi-view clustering [9,14,16,41].

Although there is no single solution to the problem of unsupervised analysis in the presence of high data dimensionality, the development of efficient clustering methods for this type of data is a great challenge for Machine Learning as it is of vital importance to obtain safer decision-making processes and better decisions that can mean greater operational efficiency, cost reduction, and risk reduction.

References

1. Achtert, E., Böhm, C., Kriegel, H.-P., Kröger, P., Zimek, A.: Robust, complete, and efficient correlation clustering. In: International Conference on Data Mining SDM, pp. 413–418 (2007)
2. Aggarwal, C.C., Procopiuc, C., Wolf, J., Yu, P.S., Park, J.-S.: Fast algorithms for projected clustering. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, pp. 61–72 (1999)
3. Aggarwal, C.C., Yu, P.S.: Finding generalized projected clusters in high dimensional space. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 70–81 (2000)
4. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pp. 94–105 (1998)
5. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data. Data Min. Knowl. Discov. **11**(1), 5–33 (2005)
6. Aristotle: Categories. In: Barnes, J. (ed.) The Complete Works of Aristotle. Translation J.L. Ackrill., vol. 2, pp. 3–24. Princeton University Press, Princeton (1995)
7. Bellman, R.: Adaptive Control Processes: A Guided Tour. Princeton University Press, Princeton (1961)
8. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell (1981)
9. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the Conference on Computational Learning Theory, pp. 92–100 (1998)
10. Böhm, C., Kailing, K., Kriegel, H.-P., Kröger, P.: Density connected clustering with local subspace preferences. In: Fourth IEEE International Conference on Data Mining, pp. 27–34 (2004)
11. Böhm, C., Kailing, K., Kröger, P., Zimek, A.: Computing clusters of correlation connected objects. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 455–466 (2004)
12. Bryan, K., Cunningham, P., Bolshakova, N.: Biclustering of expression data using simulated annealing. In: 18th IEEE Symposium on Computer-Based Medical Systems (CBMS 2005), pp. 383–388 (2005)

13. Cheng, Y., Church, G.M.: Biclustering of expression data. In: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, pp. 93–103. AAAI Press (2000)
14. Collins, M., Singer, Y.: Unsupervised models for named entity classification. In: Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, pp. 189–196 (1999)
15. Cooper, J.M., Hutchinson, D.S. (eds.): Plato: Complete Works. Hackett Publishing Co., Inc., Indianapolis (1997)
16. Dasgupta, S., Littman, M., McAllester, D.: PAC generalization bounds for co-training. Proc. Neural Inf. Process. Syst. **14**, 375–382 (2001)
17. Defays, D.: An efficient algorithm for a complete link method. Comput. J. (Br. Comput. Soc.) 20(4), 364–366 (1977)
18. Donoho, D.L.: High-dimensional data analysis: the curses and blessings of dimensionality. In: Aide-Memoire of a Lecture at the AMS Conference on Math Challenges of the 21st Century (2000)
19. Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. Proc. Nat. Acad. Sci. **95**(25), 1486–14868 (1998)
20. Filippone, M., Masulli, F., Rovetta, S., Mitra, S., Banka, H.: Possibilistic Approach to Biclustering: An Application to Oligonucleotide Microarray Data Analysis. In: Priami, C. (ed.) CMSB 2006. LNCS (LNBI), vol. 4210, pp. 312–322. Springer, Heidelberg (2006)
21. Hadamard, J.: Lectures on Cauchy's Problem in Linear Partial Differential Equations. Dover Phoenix edn. Dover Publications, New York (1923)
22. Hartigan, J.A.: Direct clustering of a data matrix. J. Am. Stat. Assoc. **67**(337), 123–129 (1972)
23. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice-Hall, Upper Saddle River (1999)
24. Kailing, K., Kriegel, H.P., Kröger, P.: Density-connected subspace clustering for high-dimensional data. In: Proceedings of the 2004 SIAM International Conference on Data Mining, pp. 246–257 (2004)
25. Koffka, K.: Principles of Gestalt Psychology. Harcourt, Brace, New York (1935)
26. Köhler, W.: Gestalt Psychology. Liveright, New York (1929)
27. Kriegel, H.-P., Kröger, P., Zimek, A.: Clustering high dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans. Knowl. Discov. Data (TKDD) **3**(1), 1–58 (2009)
28. Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms. Wiley, New York (2004)
29. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
30. Laney, D.: 3D data management: controlling data volume, velocity and variety. Gartner. http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf. Accessed 6 February 2001
31. Laney, D.: The importance of 'Big Data': a definition. Gartner. http://www.gartner.com/resId=2057415. Accessed 21 June 2012
32. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. IEEE Trans. Comput. Biol. Bioinf. **1**, 24–45 (2004)
33. Mitra, S., Banka, H.: Multi-objective evolutionary biclustering of gene expression data. Pattern Recogn. **39**(12), 2464–2477 (2006)

34. Moore, G.E.: Cramming more components onto integrated circuits. Electronics 38(8), 114–117 (1965). Reprinted in. Proc. IEEE **86**(1), 82–85 (1998)
35. Rokach, L., Maimon, O.: Clustering methods. In: Rokach, L., Maimon, O. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 321–352. Springer, USA (2005)
36. Rovetta, S., Masulli, F.: Shared farthest neighbor approach to clustering of high dimensionality, low cardinality data. Pattern Recogn. **39**, 2415–2425 (2006)
37. Sibson, R.: SLINK: an optimally efficient algorithm for the single-link cluster method. Comput. J. (Br. Comput. Soc.) 16(1), 30–34 (1973)
38. Steinbach, M., Ertoz, L., Kumar, V.: Challenges of clustering high dimensional data. In: Wille, L.T. (ed.) Proceedings of New Directions in Statistical Physics Econophysics, Bioinformatics, and Pattern Recognition, pp. 273–307. Springer, Berlin (2004)
39. Tanay, A., Sharan, R., Shamir, R.: Discovering statistically significant biclusters in gene expression data. Bioinformatics **18**, S136–S144 (2002)
40. Yang, J., Wang, H., Wang, W., Yu, P.: Enhanced biclustering on expression data. In: Proceedings of the Third IEEE Symposium on BioInformatics and Bioengineering (BIBE03), pp. 1–7 (2003)
41. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: Proceeding ACL 1995 Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics, pp. 189–196 (1995)
42. Wertheimer, M.: Untersuchungen zur Lehre von der Gestalt II. Psychologische Forschung **4**, 301–350 (1923)
43. Zhang, Z., Teo, A., Ooi, B.C., Tan, K.-L. : Mining deterministic biclusters in gene expression data. In: Proceedings of the Fourth IEEE Symposium on Bioinformatics and Bioengineering (BIBE04), pp. 283–292 (2004)

What are Clusters in High Dimensions and are they Difficult to Find?

Frank Klawonn[1,2](✉), Frank Höppner[1], and Balasubramaniam Jayaram[3]

[1] Department of Computer Science, Ostfalia University of Applied Sciences, Salzdahlumer Str. 46/48, 38302 Wolfenbuettel, Germany
{f.klawonn,f.hoeppner}@ostfalia.de

[2] Biostatistics, Helmholtz Centre for Infection Research, Inhoffen Str. 7, 38124 Braunschweig, Germany
frank.klawonn@helmholtz-hzi.de

[3] Department of Mathematics, Indian Institute of Technology Hyderabad, Yeddumailaram 502 205, India
jbala@iith.ac.in

Abstract. The distribution of distances between points in a high-dimensional data set tends to look quite different from the distribution of the distances in a low-dimensional data set. Concentration of norm is one of the phenomena from which high-dimensional data sets can suffer. It means that in high dimensions – under certain general assumptions – the relative distances from any point to its closest and farthest neighbour tend to be almost identical. Since cluster analysis is usually based on distances, such effects must be taken into account and their influence on cluster analysis needs to be considered. This paper investigates consequences that the special properties of high-dimensional data have for cluster analysis. We discuss questions like when clustering in high dimensions is meaningful at all, can the clusters just be artifacts and what are the algorithmic problems for clustering methods in high dimensions.

1 Introduction

Clustering is an exploratory data analysis method applied to data in order to discover structures or groups – called clusters – in a data set. Data objects within the same cluster should be similar, data objects from different clusters dissimilar. This means that cluster analysis must be based on a similarity or nearness concept to measure how close or similar data objects are. Often a dual concept to similarity or nearness – a distance measure – is used. Especially, for data sets having exclusively real-valued attributes, i.e. data sets that are subsets of $\mathbb{R}^m$ where m is the dimension of the data set, the Euclidean metric or – more generally – any metric derived from a norm on $\mathbb{R}^m$ can be seen as a candidate of a distance measure on which clustering can be based.

In this paper, we focus on clustering high-dimensional data having only real-valued attributes.

The term *curse of dimensionality* was coined by Bellman [1], referring to the combinatorial explosion that is often implied by handling a large number of

© Springer-Verlag Berlin Heidelberg 2015
F. Masulli et al. (Eds.): CHDD 2012, LNCS 7627, pp. 14–33, 2015.
DOI: 10.1007/978-3-662-48577-4_2

dimensions. In the context of cluster analysis, for many algorithms the number of dimensions does not cause serious computational problems. The number of data objects is usually more critical from the computational complexity point of view than the number of dimensions.

Today, the term curse of dimensionality is understood in more general terms, covering also other aspects of high-dimensional data than just problems of computational complexity. Distance measures like the Euclidean distance for high-dimensional data exhibit surprising properties that differ from what is usual for low-dimensional data. Two examples for such properties are the *concentration of norm phenomenon* – stating that under certain assumptions the relative distances from any point to its closest and farthest neighbour tend to be almost identical for high-dimensional data – and the *hubness phenomenon* where it seems that for high-dimensional data the distribution of the number of times a data point occurs among the k nearest neighbours of other data points is extremely skewed to the right and a few data points – the hubs – are found very often among the k nearest neighbours of other data points.

But what are the consequences of these effects for cluster analysis? In order to find answers to this question, we have to take a closer look at these phenomena in Sect. 2 and we also need a brief overview on clustering approaches in Sect. 3. In Sect. 4, we relate properties of high-dimensional data to clustering and discuss the crucial question what a cluster in high dimensions is. Based on these considerations, we consider the consequences for clustering high-dimensional data in terms of what is meaningful and in terms of algorithmic problems in Sect. 5. We summarise the results of the paper in Sect. 6.

2 Properties of High-Dimensional Data

The *concentration of norm phenomenon* (CoN) can formally be described in the following way [2,3]. Let X_m be an m-dimensional random vector and let $d_m(x)$ denote the distance of $x \in \mathbb{R}^m$ to the origin of the coordinate system based on a suitable distance measure, for instance the Euclidean distance. Let $n \in \mathbb{N}$ be the size of the sample that is taken from the random vector X_m. Let $d_m^{(\max)}$ and $d_m^{(\min)}$ denote the largest and the smallest distance of a point in the sample to the origin of the coordinate system. Then

$$\lim_{m\to\infty} \operatorname{Var}\left(\frac{d_m(X_m)}{\operatorname{E}(d_m(X_m))}\right) = 0 \;\Rightarrow\; \frac{d_m^{(\max)} - d_m^{(\min)}}{d_m^{(\min)}} \to_p 0 \qquad (1)$$

holds, where $\to_p$ denotes convergence in probability. In other words, when the relative variance – relative with respect to the mean distance – of the distances to the origin converges to zero for higher dimensions, then the relative difference of the closest and farthest point in the data set goes to zero with increasing dimensions. The requirement that the relative variance goes to zero is for instance satisfied when the random vector X_m is a sample from m independent and identically distributed random variables with finite expectation and variance and the Euclidean distance is used as the distance measure.

The converse theorem also holds [3].

It should be noted that the choice of the origin of the coordinate system as the query point to which the distances of the data points are computed is not of importance. Equation (1) is also valid for any other query or reference point. The same applies to the distance measure. A detailed investigation on fractional distances is provided in [4]. A discussion on various distance measures in connection with the CoN phenomenon is provided in [5–7].

The *hubness phenomenon* [8] is another property that is often observed for high-dimensional data. For the hubness phenomenon, one counts for each point of a data set how often this points occurs among the k nearest neighbours of other points where k is a fixed constant. Especially for uniformly distributed data one would expected that each point roughly occurs equally often among the k nearest neighbours of other points. However, for higher dimensions this is not true. This means that some points – called hubs – occur extremely often among the k nearest neighbours of other points.

Low et al. [9] argue that the hubness phenomenon is actually not a direct effect of high-dimensional data, but a boundary effect. The reason why it is connected to high-dimensional data is simply that for higher dimensions the proportion of the data at the boundary of the data set increases exponentially with the dimensions. Especially, when $n + 1 \leq m$ holds, where n is the number of data points and m is the number of dimensions, and all data points lie in general position, they also automatically lie on the convex of the data points.

Before we discuss what these phenomena mean for cluster analysis, we give a brief overview on basic clustering techniques in the following section.

3 Cluster Analysis

Cluster analysis aims at grouping a data set into clusters where data objects within a cluster are similar to each other and different from data objects in other clusters. Especially for data with real-valued attributes, similarity is usually defined based on a notion of distance like the Euclidean or the Manhattan distance. In order to understand how the CoN and the hubness phenomenon might affect cluster analysis, it is important to understand how these distances are used in the clustering process. Therefore, we provide a brief overview on cluster analysis which is far from being complete. For more details on cluster analysis we refer to books like [10,11].

Hierarchical agglomerative clustering is a relational clustering technique that is based on a distance matrix in which the pairwise distances of the data objects are entered. For data with real-valued attributes, these distances are often simply the Euclidean distances between the data points. Initially, each data point is considered as a separate cluster and then – step by step – the closest points or clusters are joint together to form a larger cluster until all data points end up in one cluster. The number of clusters is determined based on the dendrogram that shows in which order the clusters were joint together and how large the distances were between the joint clusters.

Hierarchical clustering cannot avoid at least quadratic complexity in the number of data points, since the required distance matrix is already quadratic in the number of data objects. Therefore, hierarchical clustering can be problematic for larger data sets. And when we deal with high-dimensional data, the data set is usually bigger, at least compared to the number of dimensions. Of course, for very high-dimensional data, this might not be the case.

When the data set to be clustered is a subset of $\mathbb{R}^m$, one would need to calculate the distance matrix for hierarchical clustering from the (Euclidean) distances between the vectors. Various clustering techniques are proposed that avoid the computation of a distance matrix that leads to quadratic complexity in the number of data objects.

It should be noted that it is usually recommended to carry out some kind of normalisation on the single dimensions to avoid that the measurement units of the single dimensions influence the distances and therefore also the clustering results. If, for instance, one of the attributes would be the height of persons, the data could be represented in various measurement units like centimetres, metres or also millimetres. The measurement of the height in metres would mean that the difference of two "data objects" (persons) in this attribute will normally be less than 1. If, however, the height is measured in millimetres, a difference of 100 between two persons would not be unusual. There are various strategies to normalise a real-valued attribute, for example z-score normalisation where the mean value is subtracted from each value and then a division by the sample standard deviation of the corresponding attribute is carried out. Of course, there might be good reason not to carry out normalisation, since normalisation will change the spatial distribution and geometrical shape of the data. But this decision depends on the specific data set and the meaning and the relations betweens its attributs. However, normalisation is not the topic of this paper and we refer to the overview on normalisation in [11].

Prototype-based clustering like the well known k-means algorithm [12] tries to avoid this complexity problem. From a purely algorithmic point of view, k-means clustering can be described as follows. First the number of clusters k must be fixed. Then each of the k clusters is represented by a prototype $v_i \in \mathbb{R}^m$ when we want to cluster m-dimensional data. These prototypes are chosen randomly in the beginning. Then each data vector is assigned to the nearest prototype (with respect to the Euclidean distance). Then each prototype is replaced by the centre of gravity of those data assigned to it. The alternating assignment of data to the nearest prototype and the update of the prototypes as cluster centres is repeated until the algorithm converges, i.e., no more changes happen.

This algorithm can also be seen as a strategy for minimising the objective function

$$f = \sum_{i=1}^{k} \sum_{j=1}^{n} u_{ij} d_{ij} \tag{2}$$

under the constraints

$$\sum_{i=1}^{k} u_{ij} = 1 \qquad \text{for all } j = 1, \ldots, n \tag{3}$$

where $u_{ij} \in \{0, 1\}$ indicates whether data vector x_j is assigned to cluster i ($u_{ij} = 1$) or not ($u_{ij} = 0$). $d_{ij} = \| x_j - v_i \|^2$ is the squared Euclidean distance between data vector x_j and cluster prototype v_i.

One of the problems of k-means clustering is that it can be quite sensitive to the choice of the initial prototypes and can easily get stuck in local minima of the objective function (2), i.e. leading to counterintuitive clustering results.

Fuzzy k-means clustering[1] [13,14] is a generalisation of k-means clustering and is less sensitive to the initialisation, since the number of local minima of the objective function can be reduced [15]. For fuzzy clustering, the constraints $u_{ij} \in \{0, 1\}$ is relaxed to $u_{ij} \in [0, 1]$. However, this change alone would still yield the same optimum of the objective function (2). Therefore, a so-called fuzzifier $w > 1$ is introduced in the objective function:

$$f = \sum_{i=1}^{k} \sum_{j=1}^{n} u_{ij}^{w} d_{ij}. \tag{4}$$

k-means and its fuzzified version can also be extended to adapt to other than spherical cluster shapes [16] or clusters of different sizes [17]. For a more detailed overview on fuzzy cluster analysis we refer to [18,19].

Gaussian mixture models can be seen as a probabilistic model for clustering where the data from each cluster are assumed to represent a sample from a multidimensional normal distribution. The expectation-maximisation algorithm (EM) can be viewed as a generalisation of the k-means algorithm where prototypes in the form the parameters of the normal distributions and assignments to clusters in terms of probabilities are estimated in an alternating fashion.

All the above mentioned clustering algorithm assume that the number of clusters is known. There are, of course, methods to estimate the number of clusters based on cluster validity measures or techniques from model selection like the Bayes information criterion. But there are also clustering techniques that automatically determine the number of clusters. These clustering algorithms are usually density-based, i.e. they try to find regions of higher data concentration that form clusters. Examples for such algorithms are DBSCAN [20], DENCLUE [21] and OPTICS [22].

Subspace clustering – for an overview see for instance [23] – refers to approaches that are especially tailored for high-dimensional data. Their main idea is not to cluster the data in all dimensions, but to look for clusters in projections to lower dimensions. These projections can be axes-parallel, but do not have to be. Essentially, subspace clustering comprises two component that are combined in the algorithm: Finding a suitable projection and applying a clustering algorithm in the corresponding subspace defined by the projection.

Before we take a closer look at high-dimensional data in connection with cluster analysis, we mention a few examples where it is of interest to cluster

[1] For fuzzy clustering, the number of clusters is usually denoted by c, so that fuzzy c-means clustering (FCM) is the common term. But for consistency reasons, we always denote the number of clusters by k.

high-dimensional data. First of all, when can we speak of high-dimensional data? The CoN phenomenon and the hubness effect can already be quite noticeable 30 dimensions. Data of this dimensionality can easily be found in many applications like industrial production where simultaneously measurements from a larger number of sensors is recorded constantly. Patient data in medicine including the laboratory results can also have a significant number of attributes.

But there are also applications having 10,000 or more dimensions. High throughput technologies like microarrays can measure the expression of far more than 10,000 genes easily. Often, the genes are clustered, which would mean that the number of data objects exceeds 10,000, but not the number of dimensions. But sometimes it is also interesting to cluster microarray data from different experiments or different individuals which leads to such high-dimensional data, since then each gene or its expression value is considered as an attribute (see for instance [24]). Other high throughput technologies like mass spectrometry for proteomics and in combination with gas chromatography for metabolomics or next generation sequencing also produce data with thousands of dimensions. In [25] growth curves of more than 4000 mutants under more than 100 conditions were measured by the VITEK®2 technology and clustered.

So there is an obvious need to cluster data with more than 30 up to tens of thousands dimensions. But what are clusters in such high dimensions? How do they look like and what can we expect from them? The following section will discuss these question in more detail.

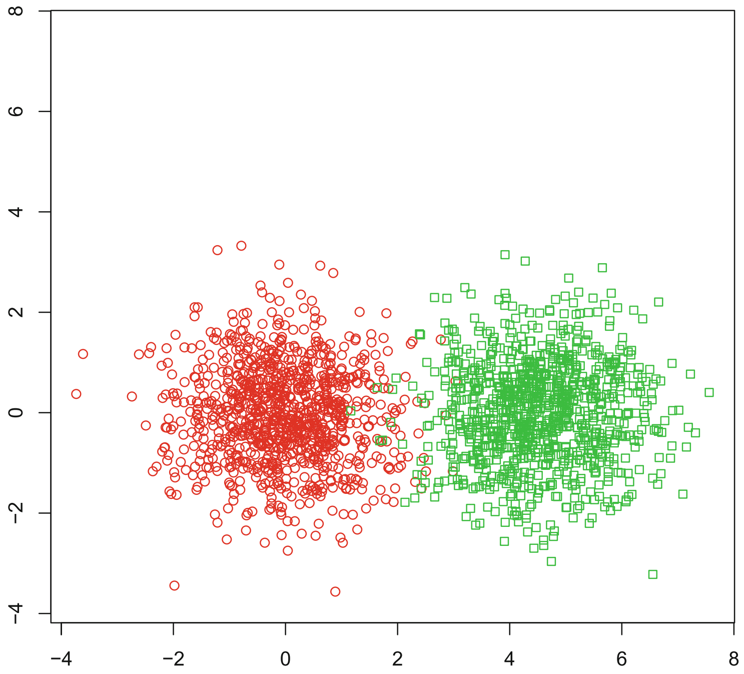


Fig. 1. An artificial data set with two clusters characterised by the x-values.

4 What are Clusters, Especially in Higher Dimensions?

The CoN phenomenon affects high-dimensional data in general. Especially, nearest neighbour search and database queries of the top k similar cases might not be meaningful anymore for high-dimensional data [26].

But what are the consequences for cluster analysis which is usually also based on distance notions? In order to discuss these consequences, it is essential to first clarify what we mean by or what we expect from a cluster in high dimensions. However, to specify what we expect from a high-dimensional cluster is not so obvious as the following two artificial examples show. Both examples contain two clusters, each of them having 1000 data objects.

The first example is essentially a one-dimensional example. Both clusters are generated by normal distributions with variance 1. But the first cluster has expected value 0, whereas the second cluster has expected value 4.5. In order to make it a high-dimensional example, we add additional dimensions that do not contribute to the distinction of the clusters. For both clusters, the values of the additional dimensions come from standard normal distributions, i.e. with expected value 0 and variance 1.

Figure 1 shows the two-dimensional data set. The relevant attribute to distinguish the two clusters is shown on the x-axis, whereas the first irrelevant attribute is shown on the y-axis.

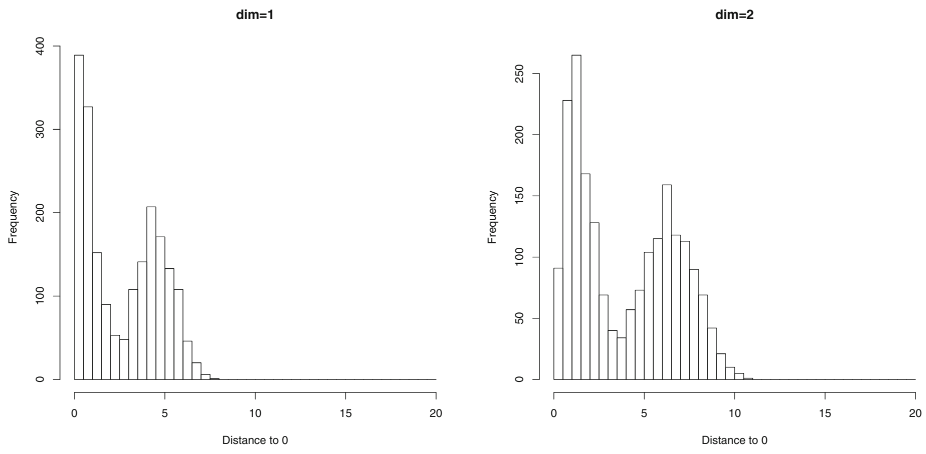


Fig. 2. Distribution of the distances of the data to the origin of the coordinate system in the first example for one (left) and two (right) dimensions.

We now consider the distribution of the Euclidean distances of the points in the data set to the origin of the coordinate system. Figure 2 shows this distribution for the one-dimensional case on the left hand side when only the relevant attribute is considered. The histogram has one peak for each cluster. The first cluster is represented by the peak at 1. This corresponds to the data from the standard normal distribution scattering around zero with an average distance of 1, since the variance is 1. The second peak is at 4.5, the expected value of the second normal distribution.

The right-hand side of Fig. 2 shows the distribution of the distances in the two-dimensional case, i.e. when in addition to the relevant attribute, one irrelevant attribute is present that does not contribute to the distinction of the clusters.

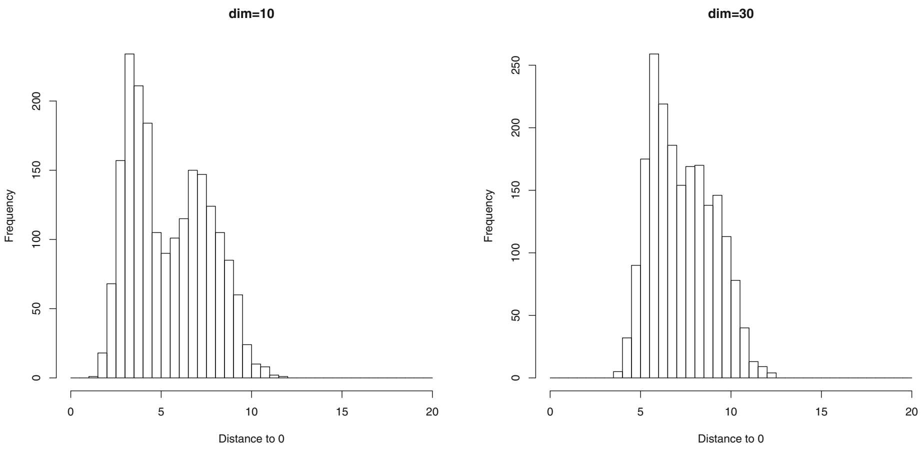


Fig. 3. Distribution of the distances of the data to the origin of the coordinate system in the first example for 10 (left) and 30 (right) dimensions.

When we increase the number of attributes whose distributions are the same for both clusters, the two peaks in the histograms in Fig. 2 start to melt together, as can be seen in Fig. 3 which shows the histograms for 10 and 30 dimensions. For 10 dimensions, two peaks can still be identified, whereas they are almost invisible in 30 dimensions.

For 50 dimensions in Fig. 4 on the left-hand side, only one peak can be identified although the distribution is still skewed. For 200 dimensions in Fig. 4 on the right-hand side even the skewness has vanished.

Table 1 shows some statistical characteristics of the intra- and the inter-cluster distances for this example. The intra-cluster distances are obtained by computing the Euclidean distance between each pair of points from the same cluster. For the inter-cluster distances, the two points are always chosen from different clusters. For low dimensions, the distributions of the intra- and the inter-cluster distances differ significantly. But with an increasing number of dimensions, these two distributions become more and more similar.

What does this simple artificial example tell us about clusters in high dimensions? There is no doubt about the two clusters in low dimensions as Fig. 1

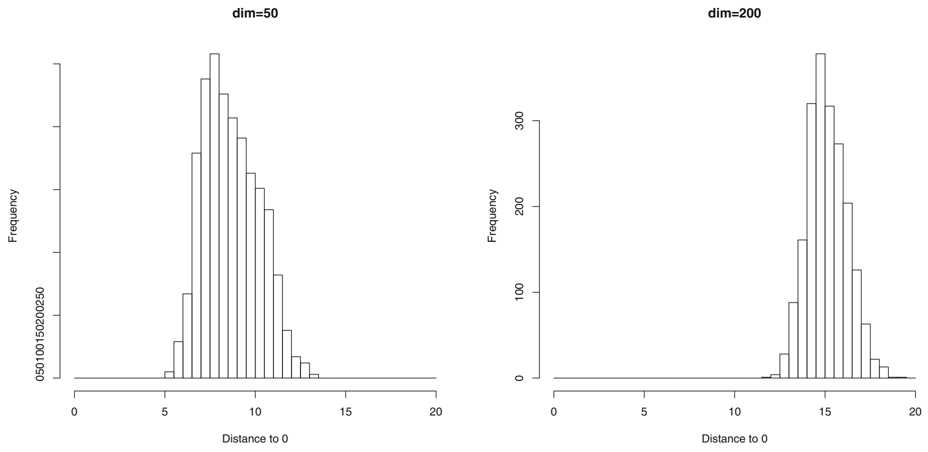


Fig. 4. Distribution of the distances of the data to the origin of the coordinate system in the first example for 50 (left) and 200 (right) dimensions.

illustrates it clearly for two dimensions and as the histograms indicate it with their two peaks in low dimensions. But are these two clusters still present in high-dimensions? Neither the distance histograms for higher dimensions nor the vanishing difference between the intra- and the inter-cluster distance distributions indicate that there are two clusters present.

The histograms produced for this example are all based on the Euclidean distance or norm. François et al. demonstrated in [4] that the concept of L^p-norms can help to relax the effects of the curse of dimensionality. The L^p-norm is defined as

$$\| x \|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{\frac{1}{p}} .$$

Choosing $p = 2$ leads to the Euclidean norm. But other values for p are also possible. For $p \geq 1$ these norms are called L^p- or Minkowski norms. For $0 < p < 1$ they are called fractional norms although they do not satisfy the properties of a norm. Both [4,27] make a strong point for the use of fractional norms or metrics for high-dimensional data. However, for the simple example considered here, fractional norms do not improve the situation. In contrast, a high value of p in the L^p-norm will lead to better results here. The two clusters start to melt together later for L^p-norms with large p. Figure 5 corresponds to Fig. 4, except that the L^p-norm with $p = 400$ is used. The clusters are now not fused together completely, even for 200 dimensions.

Before we further discuss this example, we consider another artificial example. Again, there are two clusters. In contrast to the first example, there is not a single attribute by which the two clusters can be distinguished, but each attribute contributes a little bit to the distinction of the two clusters. For each attribute, we assume a normal distribution with variance 1 for both clusters, but with

Table 1. Statistical characteristics of the intra- and inter-cluster distances for the first example.

Dimensions										
intra/**inter**	1	2	3	5	10	20	30	50	100	200
Minimum	< 0.001	0.001	0.026	0.237	0.871	2.479	3.782	6.031	9.841	15.967
Minimum	< 0.001	0.048	0.129	0.384	1.733	3.318	4.732	6.336	10.824	16.613
5 % quantile	0.088	0.537	1.053	1.860	3.269	5.136	6.649	8.882	12.954	18.771
5 % quantile	2.218	3.377	3.782	4.343	5.390	6.925	8.193	10.192	13.995	19.560
25 % quantile	0.450	1.283	1.969	2.877	4.309	6.167	7.660	9.884	13.932	19.752
25 % quantile	3.607	5.278	5.597	6.064	6.962	8.334	9.515	11.431	15.110	20.626
Median	0.954	2.004	2.772	3.699	5.107	6.931	8.394	10.602	14.624	20.446
Median	4.561	6.600	6.883	7.306	8.107	9.379	10.496	12.340	15.925	21.390
Mean	1.132	2.161	2.895	3.788	5.169	6.972	8.422	10.625	14.635	20.452
Mean	4.549	6.595	6.892	7.324	8.144	9.426	10.544	12.379	15.959	21.405
75 % quantile	1.633	2.869	3.687	4.602	5.965	7.733	9.152	11.340	15.329	21.143
75 % quantile	5.501	7.913	8.176	8.558	9.285	10.467	11.521	13.285	16.773	22.167
95 % quantile	2.788	4.314	5.157	6.022	7.281	8.943	10.295	12.439	16.349	22.158
95 % quantile	6.844	9.795	10.036	10.383	11.025	12.091	13.061	14.696	18.041	23.306
Maximum	6.990	9.952	10.046	10.490	11.683	12.914	13.942	15.943	19.802	25.211
Maximum	10.420	14.741	14.753	15.173	16.061	17.083	17.452	18.608	22.115	27.236

expected value 0 for the first, and expected value 1 for the second cluster. Again, each cluster contains 1000 data objects.

Figure 6 shows the data for two dimensions. The two clusters are visually indistinguishable. Only the use of different symbols for the two clusters provides visual information about the presence of two clusters.

In the same fashion as for the first example, Figs. 7, 8 and 9 show the distributions of the distances to the origin of the coordinate system for the second example for different dimensions. We can observe exactly the opposite effect as in the first example. For low dimensions, there is only one peak and the two clusters cannot be detected by looking at the histograms. For about 30 dimensions, the single peak starts to be separated into two peaks and the separation of the two peaks is almost perfect for 200 dimensions.

Table 2 contains the characteristics of the inter-cluster distances for the second example. The intra-cluster distances have, of course, the same characteristics as in the first example, i.e. the values correspond to the ones in Table 1 (the non-bold face rows). It is no surprise that we can observe the opposite effect as in the first example. For low dimensions, the distributions of the intra- and the inter-cluster distances are quite similar, but with an increasing number of dimensions, they become more and more distinguishable.

5 Consequences for Clustering Algorithms

The two examples in the previous section illustrate how the CoN phenomenon influences cluster analysis in high dimensions. If we assume that the clusters

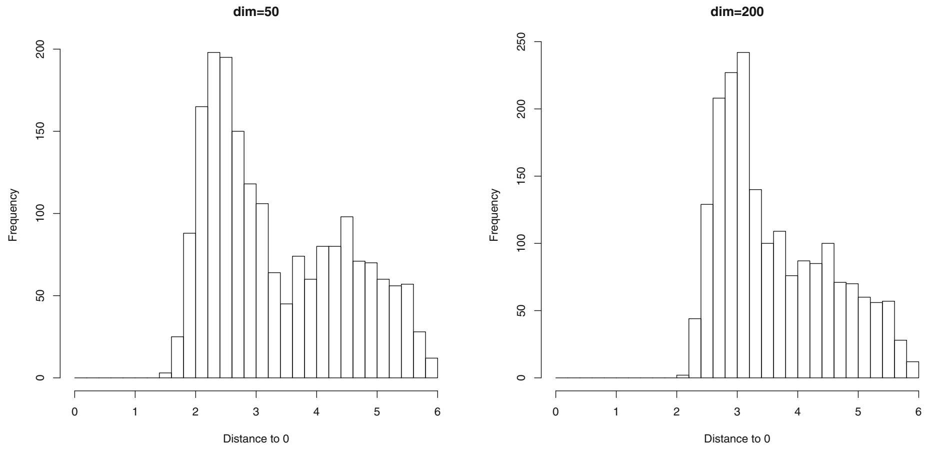


Fig. 5. Distribution of the distances of the data to the origin of the coordinate system in the first example for 50 (left) and 200 (right) dimensions based on the L^p-norm with $p = 400$.

are defined by only a few variables and most of the other variables have no connection to the clusters at all, then the CoN phenomenon destroys the distinguishability of clusters as demonstrated by the first example. But when (almost) all attributes contribute at least a little bit to the distinction of the clusters, the CoN phenomenon even helps to identify the clusters.

How many "noise" dimensions can be tolerated in the first example, until we say that there are no longer two clusters? How many dimensions do we need in the second example, until we can really speak of two clusters?

Of course, for real world data, the situation is usually more complicated. There might be some attributes that lead to good clusters and other attributes which are simply "noise" in terms of the clustering. And in contrast to the two examples, attributes might also be correlated. The first example has shown that a limited number of "noise" attributes or "noise" dimensions can be tolerated.

Table 2. Statistical characteristics of the inter-cluster distances for the second example.

Dimensions										
	1	2	3	5	10	20	30	50	100	200
Minimum	< 0.001	0.004	0.026	0.210	1.117	2.656	4.680	7.195	12.355	19.791
5 % quantile	0.114	0.687	1.340	2.382	4.082	6.385	8.212	10.984	15.998	23.112
25 % quantile	0.577	1.623	2.481	3.611	5.327	7.612	9.428	12.168	17.165	24.243
Median	1.208	2.513	3.467	4.588	6.267	8.517	10.301	13.016	17.988	25.037
Mean	1.398	2.672	3.587	4.679	6.325	8.561	10.330	13.039	17.998	25.044
75 % quantile	2.025	3.548	4.560	5.648	7.258	9.462	11.203	13.881	18.819	25.836
95 % quantile	3.341	5.206	6.260	7.288	8.765	10.882	12.540	15.178	20.031	27.001
Maximum	7.284	10.488	10.747	13.522	14.648	16.883	17.821	19.528	24.085	31.059

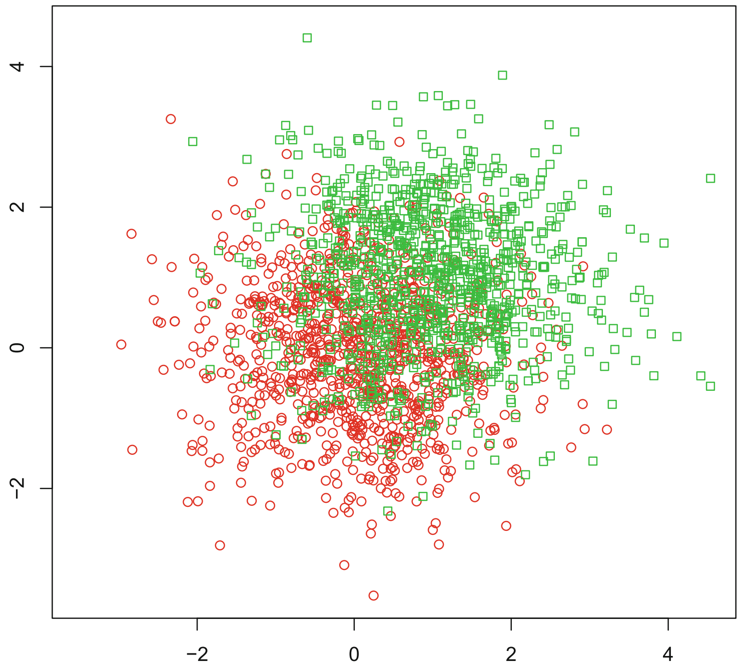


Fig. 6. An artificial data set with two clusters that are difficult to distinguish in low dimensions.

But what happens when there are different combinations of attributes, leading to different clustering results? Which clusters should we trust when there are completely different clusters for different combinations of attributes, i.e. in different subspaces?

There is, of course, a clear argument for some kind of subspace clustering approach. Typically, the number of clusters we expect or we are searching for in a data set, is limited. In many applications, even 20 clusters is already a very large number. When we have k clusters and think in terms of prototype-based clustering, the cluster centres lie in an at most $(k-1)$-dimensional hyperplane. So this means that, at least when the clustering result is good, the clustering could have been carried out on the projection of the data to this $(k-1)$-dimensional hyperplane.

The idea of finding interesting patterns in lower dimensions in high-dimensional data sets by suitable projections is well known in data analysis as a technique called projection pursuit [28]. However, projection pursuit focuses mainly on two- and three-dimensional projections for visualisation purposes and does not specifically aim at finding clusters. Since projections of high-dimensional data to low dimensions tend to resemble normal distributions, one way to carry out projection pursuit is to generate random two- or three-dimensional projections and to apply a test for normality. Those projections for which the normality

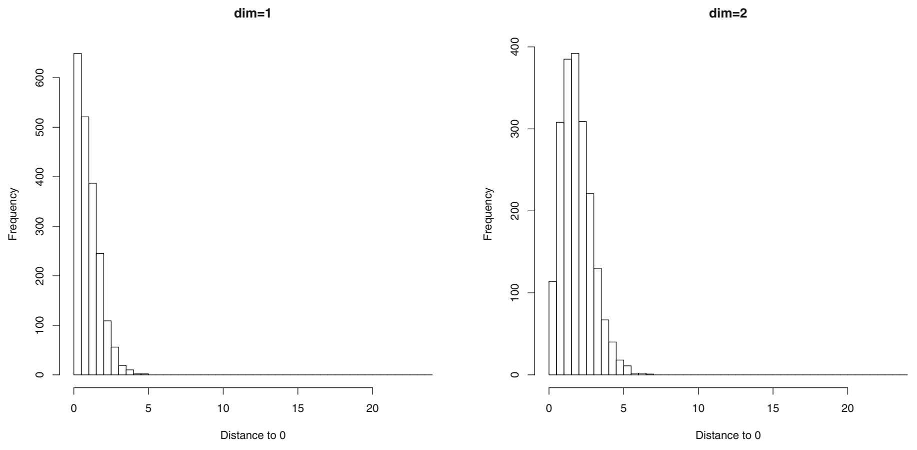


Fig. 7. Distribution of the distances of the data to the origin of the coordinate system in the second example for one (left) and two (right) dimensions.

hypothesis is rejected are then presented to the user for further inspection or the projections can be ranked according to their p-values that the tests for normality yielded. However, a deviation from a normal distribution does not necessarily indicate that there are clusters in the data set.

Finding the right projection is a crucial problem in subspace clustering. It can be seen as a feature selection strategy [29] when only axes-parallel projections are considered.

It should be taken into account that we have to deal with the problem of multiple testing for very high-dimensional data like in the case of microarrays: multiple testing in the sense that we test a larger number of projection for the presence or absence of clusters. When we have many attributes, the number of possible projections is extremely large – or even infinite if we drop the restriction to axes-parallel projections. This means that the probability that a certain projection contains clusters just by chance might not be negligible.

As an example, consider a data set with 200 data objects and 10,000 attributes which are all independent samples from a uniform distribution on the unit interval. So we have 200 uniformly distributed data points in the unit hypercube of dimension 10,000. There are, of course, no clusters. But we do not know this fact, since in reality we would not know from which distribution the data were generated. In order to find clusters in subspaces, we only consider projections to three dimensions here. Figure 10 illustrates a projection where we can see two clusters. One cluster is above the diagonal plane of the unit cube, the other one below.

What is the chance that we can find such a projection for our uniformly distributed data from the 10,000-dimensional unit hypercube? Assume the separation between the clusters should be at least 0.1 units, i.e. the plane in Fig. 10 would be turned into a box of height[2] 0.1. What is the probability that no

[2] In Fig. 10 the separation between the two clusters is chosen larger for illustration purposes.

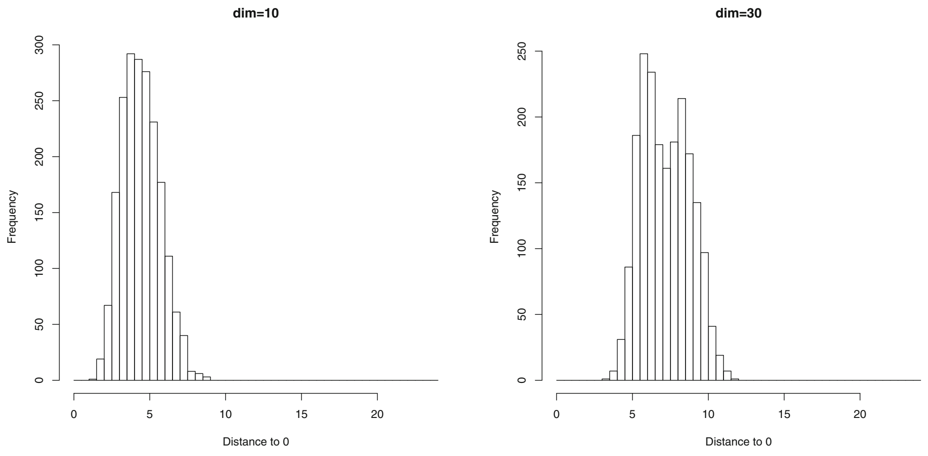


Fig. 8. Distribution of the distances of the data to the origin of the coordinate system in the second example for 10 (left) and 30 (right) dimensions.

projection will look like this, i.e. that each projection contains at least one point within the separating box? The box has a volume of

$$\sqrt{1^2 + \left(\frac{1}{2}\right)^2} \cdot \sqrt{1^2 + \left(\frac{1}{2}\right)^2} \cdot 0.1 \;=\; 0.125,$$

so that the remaining volume of the cube is $1 - 0.125 = 0.875$. The probability that all points lie in this remaining cube outside the separating box, i.e. that we have two artificial clusters, is 0.875^{200} and the probability that these artificial random cluster do not occur for a single projection to three dimensions is $1 - 0.875^{200}$. The probability that we will not find such random clusters in any of the $10000 \cdot 9999 \cdot 9998$ possible projections to three dimensions is

$$\left(1 - 0.875^{200}\right)^{10000 \cdot 9999 \cdot 9998} \;\approx\; 0.08.$$

So there is only an 8 % chance that we will not find these spurious random clusters in any of the three-dimensional projections of our data set from a high-dimensional uniform distribution. It should be noted that we have only considered projections to three dimensions and a specific separating plane between the clusters. If we allow projections to more than three dimensions and take into account that the random clusters might be separated by other planes or geometric shapes, the situation gets much worse and the probability that we find random clusters in a projection is almost 1, even if we have more than 200 data points.

Therefore, it is highly recommended to carry out some additional tests to verify whether clusters found by subspace clustering in very high-dimensional data set are not just random effects. One can apply a permutation test in which the values in each column of the data table are randomly permuted. Then the

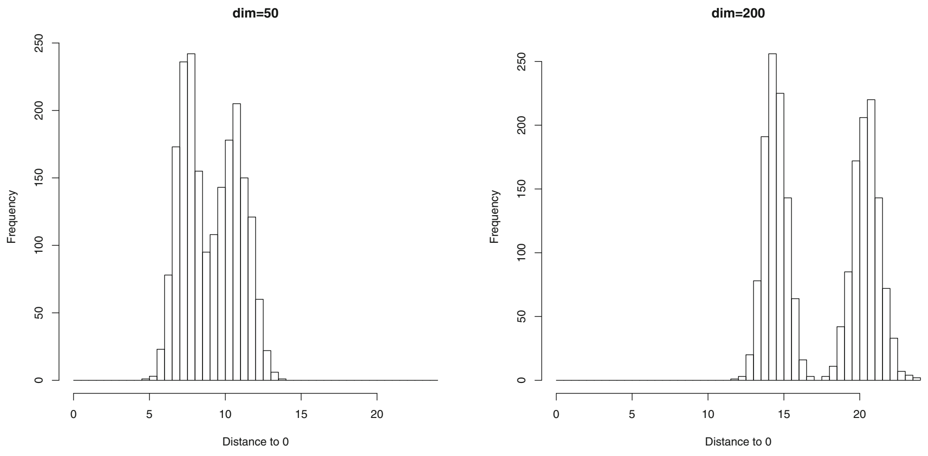


Fig. 9. Distribution of the distances of the data to the origin of the coordinate system in the second example for 50 (left) and 200 (right) dimensions.

clustering algorithm is applied to the randomly permuted data. When the clustering algorithm still finds clusters in this permuted data set, one cannot trust the clusters that were found in the original data set. The permutation test should be carried out more than once.

A Monte Carlo test is also possible by generating random data of the same dimension and with the same number of data objects, but using a distribution where no clusters should be found like the uniform distribution in the unit hypercube that we have considered above. When the algorithm can find clusters in such a random data set, the clusters in the real data set might again be just random artifacts.

Subspace clustering is definitely needed when we assume that our data set is more like the first example described in Sect. 4, i.e. there are a few attributes that contribute to the clusters and the large majority of attributes is just "noise". In this case, clustering algorithms taking all attributes into account, would have little or no chance to discover the clusters. The CoN and the hubness phenomenon will hide the clusters.

But how is the situation when the data set is more of the type as the second example described in Sect. 4, i.e. most of the attributes contribute a little bit to the clusters and only a few "noise" attributes might be present? Fig. 9 indicates that the CoN phenomenon can even make it easier to find the clusters. The reason is that the CoN phenomenon is not applicable to the whole data set. The assumption that the relative variance goes to zero is not satisfied here. The relative variance in this simple example is strictly positive due to the two clusters whose centres have a distance of $\sqrt{m}$. The expected distance to the origin is $\frac{\sqrt{m}}{2}$. So the relative variance is roughly $\frac{1}{4}$. The CoN phenomenon does occur, but in each cluster separately as can be seen from Fig. 9 where the two peaks in the

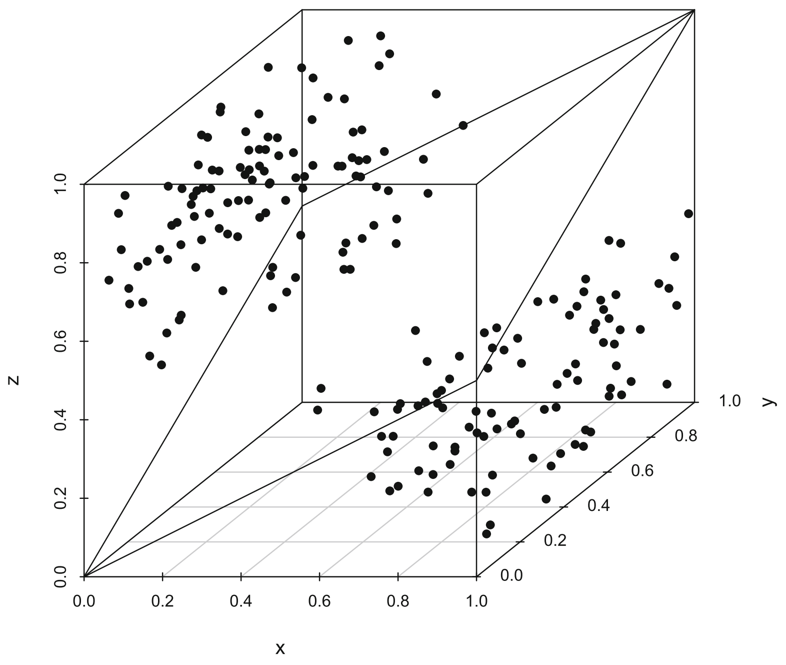


Fig. 10. A possible projection which contains clusters only by chance.

histograms for the clusters become not only better separated, but also quite narrow for high dimensions.

However, the situation is not as positive and simple as it seems. When we look at prototype-based clustering, it is not a problem of the objective function. For an ideal data set as the second example in Sect. 4, the objective function will have a clear global minimum at the centre of the two clusters. But when we have more than just two clusters, it becomes a problem of local minima of the objective function and therefore a problem of a good initialisation. Since the location of the cluster centres is not known a priori, the initial prototypes are placed “somewhere” and then suffer from the CoN phenomenon on the level of the clusters in the sense that all clusters have roughly the distance to them unless a prototype is located close to a true cluster centre.

In [30] it was demonstrated that k-means clustering has difficulties to find clusters in high dimensions, even when the clusters are well separated. One might suspect that this is the well-known sensitivity of k-means clustering to the initialisation and that fuzzy clustering might yield better results. But the contrary is the case. For fuzzy clustering, all or most of the prototypes tend to converge in the centre of gravity of the whole data set.

What is the reason for this surprising result? Fig. 11 from [30] explains this effect. It shows the objective function (4) of fuzzy clustering reduced to one parameter for a specific data set. The data set consists of a fixed number of

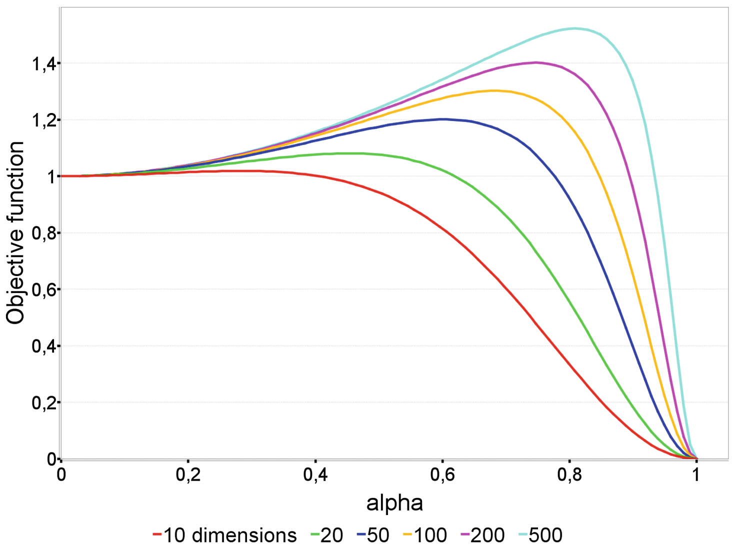


Fig. 11. The objective function of fuzzy clustering has a local minimum in the centre of all data points for high-dimensional data.

well-separated clusters – each of them concentrated in a single point – distributed uniformly on the surface of an $(m-1)$-dimensional unit hypersphere. The cluster prototypes are first all placed into the origin, i.e. the centre of gravity of all the data points. Then the cluster prototypes are moved along the lines connecting each cluster prototype with one of the true cluster centres. So at 0 on the x-axis in Fig. 11 all prototypes are at the origin (radius=0), at 0.5 they are halfway between the origin and the true cluster centres and at 1 each of the prototypes is placed exactly in one of the cluster centres. As can be seen from the figure, the clear global minimum of the objective function is at 1, i.e. when all prototypes are placed in the true cluster centres. But there is a local minimum at the origin, separated by a local maximum from the global minimum. The local maximum is shifted more to the right for higher dimensions. Since the algorithm to minimise the objective function of fuzzy clustering can be view as a gradient descent technique [31], the cluster prototypes will end up in the local minimum at the origin when the initialisation is not close enough to the true cluster centres.

These considerations about the objective function for fuzzy clustering in high dimensions demonstrate that the CoN phenomenon occurs on the level of the cluster centres in the sense that all cluster centres have roughly the same distance from a prototype that is not placed close to a true cluster centre. So this is not a problem of the objective function which clearly shows a global minimum at the correct cluster centres. It is a problem of the optimisation algorithm. In contrast to this example where the clusters are well separated, the first example in Sect. 4 would also cause a problem for the objective function, since the minimum would not be pronounced very clearly.

There are ways to partly avoid these problems. One way is to try to adjust the fuzzifier w in the objective function (4) depending on the number of dimensions. The higher the number of dimensions, the smaller, but of course larger than 1, the fuzzifier should be chosen. A better way to avoid the tedious adjustment of the fuzzifier is to use a polynomial fuzzifier function [32] that replaces the power function u^w by a quadratic polynomial of the form $w \cdot u^2 + (1 - w) \cdot u$ with $u \in [0, 1]$. This leads to a convex combination of the standard k-means clustering objective function (2) and the objective function for fuzzy clustering (4) with fuzzifier 2.

Density-based clustering suffers from the problem that density can vary to an extreme extend in high dimensions and it is very difficult to adjust the parameter settings.

6 Conclusions

Clustering high-dimensional data is a difficult task. The problem starts already with the understanding of how a cluster should be characterised. Will only a few dimensions or attributes contribute to the clustering and the large majority of attributes is considered as "noise"?

We have seen with the first example in Sect. 4 that a limited number of "noise" attributes can be tolerated without destroying the clusters. If the data set is expected to contain too many "noise" attributes that are irrelevant for the clustering, subspace clustering techniques are needed. Subspace clustering drastically increases the complexity of clustering, since not only clusters need to be found but also the right subspace. Apart from this, for very high-dimensional data, subspace clustering might lead to finding spurious clusters that "look good", but are just random effects as we have illustrated in Sect. 5. Therefore, validation techniques like permutation or Monte Carlo tests should be applied to get an idea for the chances of finding spurious clusters.

If the number of "noise" attributes is limited, subspace clustering might not be necessary. However, clustering algorithms might still suffer from the CoN phenomenon. As we have seen, if the clusters are well separated, this is not a problem of the cluster model – the objective function – but a problem of the algorithm to find the best cluster model or to minimise the objective function. As for subspace clustering, it is highly recommended to evaluate the clusters. However, here other methods than permutation or Monte Carlo tests might be better suited like cross-validation in the sense of resampling [33] or the application of other cluster validity measures. A good overview on cluster validity measures can be found in [34]. These techniques are also often used to determine the number of clusters.

Missing values are a problem that is usually not considered in cluster analysis [35]. But missing values will occur with larger probability in high-dimensional data. This is still an open problem.

References

1. Bellmann, R.: Adaptive Control Processes: A Guided Tour. Princeton University Press, Princeton (1961)
2. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is nearest neighbor meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
3. Durrant, R.J., Kabán, A.: When is 'nearest neighbour' meaningful: a converse theorem and implications. J. Complex. **25**(4), 385–397 (2009)
4. François, D., Wertz, V., Verleysen, M.: The concentration of fractional distances. IEEE Trans. Knowl. Data Eng. **19**(7), 873–886 (2007)
5. Aggarwal, C.C.: Re-designing distance functions and distance-based applications for high dimensional data. SIGMOD Rec. **30**(1), 13–18 (2001)
6. Hsu, C.M., Chen, M.S.: On the design and applicability of distance functions in high-dimensional data space. IEEE Trans. Knowl. Data Eng. **21**(4), 523–536 (2009)
7. Jayaram, B., Klawonn, F.: Can unbounded distance measures mitigate the curse of dimensionality? Int. J. Data Min. Model. Manag. **4**, 361–383 (2012)
8. Radovanović, M., Nanopoulus, A., Ivanović, M.: Hubs in space: popular nearest neighbors in high-dimensional data. Mach. Learn. Res. **11**, 2487–2531 (2010)
9. Low, T., Borgelt, C., Stober, S., Nürnbberger, A.: The hubness phenomenon: fact or artifact? In: Borgelt, C., Ángeles Gil, M., Sousa, J., Verleysen, M. (eds.) Towards Advanced Data Analysis by Combining Soft Computing and Statistics, pp. 267–278. Springer, Berlin (2013)
10. Evertt, B., Landau, S.: Cluster Analysis, 5th edn. Wiley, Chichester (2011)
11. Berthold, M., Borgelt, C., Höppner, F., Klawonn, F.: Guide to Intelligent Data Analysis: How to Intelligently Make Sense of Real Data. Springer, London (2010)
12. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. Wiley, New York (1973)
13. Dunn, J.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Cybern. Syst. **3**(3), 32–57 (1973)
14. Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
15. Jayaram, B., Klawonn, F.: Can fuzzy clustering avoid local minima and undesired partitions? In: Moewes, C., Nürnberger, A. (eds.) Computational Intelligence in Intelligent Data Analysis, pp. 31–44. Springer, Berlin (2012)
16. Gustafson, D., Kessel, W.: Fuzzy clustering with a fuzzy covariance matrix. In: IEEE CDC, San Diego, pp. 761–766 (1979)
17. Keller, A., Klawonn, F.: Adaptation of cluster sizes in objective function based fuzzy clustering. In: Leondes, C. (ed.) Intelligent Systems: Technology and Applications. Database and Learning Systems, vol. IV. CRC Press, Boca Raton (2003)
18. Bezdek, J., Keller, J., Krishnapuram, R., Pal, N.: Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. Kluwer, Boston (1999)
19. Höppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. Wiley, Chichester (1999)
20. 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 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), pp. 226–231. AAAI Press (1996)
21. Hinneburg, A., Gabriel, H.H.: Denclue 2.0: fast clustering based on kernel density estimation. In: Proceedings of the 7th International Symposium on Intelligent Data Analysis, pp. 70–80 (2007)

22. Ankerst, M., Breunig, M., Kriegel, H.P., Sander, J.: Optics: ordering points to identify the clustering structure. In: Proceedings of ACM SIGMOD 1999, pp. 49–60. ACM Press (1999)
23. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans. Knowl. Discov. Data **3**(1), 1–58 (2009)
24. Kerr, G., Ruskin, H., Crane, M.: Techniques for clustering gene expression data. Comput. Biol. Med. **38**(3), 383–393 (2008)
25. Pommerenke, C., Müsken, M., Becker, T., Dötsch, A., Klawonn, F., Häussler, S.: Global genotype-phenotype correlations in pseudomonas aeruginosa. PLoS Pathogenes **6**(8) (2010). doi:10.1371/journal.ppat.1001074
26. Hinneburg, A., Aggarwal, C., Keim, D.: What is the nearest neighbor in high dimensional spaces? In: Abbadi, A.E., Brodie, M.L., Chakravarthy, S., Dayal, U., Kamel, N., Schlageter, G., Whang, K.Y. (eds.) VLDB, pp. 506–515. Morgan Kaufmann, San Francisco (2000)
27. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, p. 420. Springer, Heidelberg (2000)
28. Cook, D., Buja, A., Cabrera, J.: Projection pursuit indices based on orthonormal function expansion. J. Comput. Graph. Stat. **2**, 225–250 (1993)
29. Wang, S., Zhu, J.: Variable selection for model-based high-dimensional clustering and its application to microarray data. Biometrics **64**, 440–448 (2008)
30. Winkler, R., Klawonn, F., Kruse, R.: Fuzzy c-means in high dimensional spaces. Fuzzy Syst. Appl. **1**, 1–17 (2011)
31. Höppner, F., Klawonn, F.: A contribution to convergence theory of fuzzy c-means and its derivatives. IEEE Trans. Fuzzy Syst. **11**, 682–694 (2003)
32. Klawonn, F., Höppner, F.: What is fuzzy about fuzzy clustering? understanding and improving the concept of the fuzzifier. In: Berthold, M.R., Lenz, H.J., Bradley, E., Kruse, R., Borgelt, C. (eds.) Advances in Intelligent Data Analysis, vol. V, pp. 254–264. Springer, Berlin (2003)
33. Borgelt, C.: Resampling for fuzzy clustering. Int. J. Uncertainty Fuzziness Knowl. Based Syst. **15**, 595–614 (2007)
34. Borgelt, C.: Prototype-based Classification and Clustering. Habilitation thesis, Otto-von-Guericke-University Magdeburg (2006)
35. Himmelspach, L., Conrad, S.: Clustering approaches for data with missing values: comparison and evaluation. ICDIM **2010**, 19–28 (2010)

Efficient Density-Based Subspace Clustering in High Dimensions

Ira Assent(✉)

Department of Computer Science, Aarhus University, Aarhus, Denmark
ira@cs.au.dk
http://cs.au.dk/~ira

Abstract. Density-based clustering defines clusters as dense areas in feature space separated by sparsely populated areas. It is known to successfully identify clusters of arbitrary shapes even in noisy data. Today, we face increasingly high-dimensional data, i.e. data objects described by many attributes. Effects attributed to the "curse of dimensionality" mean that in high-dimensional spaces, traditional clustering methods fail to identify meaningful clusters. In little more than a decade, the research field of subspace clustering has established methods for identifying clusters in subsets of the attributes in such high-dimensional spaces. As the number of possible subsets is exponential in the number of attributes, efficient algorithms are crucial. This short survey discusses challenges in this area, and presents models and algorithms for efficient and scalable density-based subspace clustering.

Keywords: Data mining · Clustering · Density-based clustering · Subspace clustering · High dimensional · Efficiency

1 Introduction

With increasing availability for sensors, large amounts of data are being collected in a variety of applications. Ranging from science to business and end user applications, the volume of data collections is growing continuously. At the same time, the number of attributes describing the data is increasing as well. As prices for storage are dropping, many features are routinely collected for a given data object.

Knowledge discovery in databases is a well-established approach for making sense of large data collections in a (semi-)automatic fashion. A core step in the process is the extraction of patterns from task-relevant cleaned data using data mining algorithms. A popular data mining method is clustering, which aims to discover groups of data objects based on mutual similarity.

Different clustering paradigms exist in the clustering literature. Density-based clustering defines clusters as maximal groups of density-connected objects that are separated by sparsely populated areas in feature space [8]. This approach is known to perform well even in data collections including noise.

In high-dimensional data, i.e., in data with a large number of attributes (or dimensions or features), clustering is known to fail due to effects attributed to the "curse of dimensionality" [6]. As the number of dimensions increases, the distances between objects in the feature space become more and more alike [7]. As a consequence, the distance in the high-dimensional space no longer discerns between nearby and far away objects. Since dense objects are defined as those with at least a certain number of objects within their neighborhood, this means that density-based clustering in high-dimensional data fails to detect meaningful clusters.

In recent years, the data mining community has researched subspace clustering methods as a method for clustering in high dimensional data. The general idea is to identify clusters in subspace projections of the data. This is based on the important observation that relevance of dimensions might not be globally uniform; instead, locally different subspace projections might be relevant for different clusters. In this manner, it is also possible that a data object is assigned to different clusters in different subspace projections.

In this short survey, we give a brief overview over density-based subspace clustering, and we review some of our work in this area. For a more general discussion of density-based clustering, clustering in high-dimensional data, and subspace clustering also using other clustering paradigms, the interested reader is referred to a number of recent surveys and evaluation studies on the topic [2,11,12,14,17–19].

2 Density-Based Subspace Clustering

Density-based clustering was originally introduced as DBSCAN in the seminal work in [8]. A cluster is defined as a maximal group of density-connected objects that originate around one or more dense *core* objects. Objects are core objects when their density, i.e., the number of objects within a neighborhood, exceeds a pre-defined threshold.

Definition 1. ***Density-Based Cluster.*** *A density-based cluster C w.r.t. density threshold $minPoints$ and neighborhood radius ε is defined as follows:*

- *All objects in C are dense, i.e., there are at least $minPoints$ objects in their neighborhood:*

$$\forall o \in C:\ |N_\varepsilon(o)| = |\{p \in DB \mid dist(p,o) \leq \varepsilon\}| \geq minPoints$$

- *All objects are density-connected, i.e., for any two objects in the cluster, there is a chain of objects between them such that the successor in the chain is in the predecessor's neighborhood:*

$$\forall o, p \in C : \exists\ q_1, \ldots, q_m \in C : \forall_{i \in \{2,\ldots,m\}}\ q_i \in N_\varepsilon(q_{i-1})\ \wedge q_1 = o\ \wedge q_m = p$$

– *The cluster is maximal, i.e., any two density-connected objects are either both in cluster C or both not in it:*

$$\forall o, p \in DB : o, p \text{ density-connected } \Rightarrow (o \in C \Leftrightarrow p \in C)$$

This definition is illustrated in Fig. 1. The density of an object is assessed by counting the number of objects in its neighborhood, i.e., within ε distance. If the density exceeds the minPoints threshold, the object is a core object. From such core objects, all objects that are density-connected are assigned to the same cluster. Sparse areas in feature space separate a cluster (points to the lower left) from another cluster (points to the upper right). Since not all objects are necessarily assigned to a cluster, density-based clustering tolerates noise in the data.

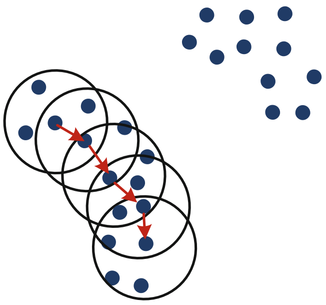


Fig. 1. Density-based clustering: a cluster is a maximal group of objects that are connected via mutual inclusion in their neighborhood starting from at least one dense core object.

In high-dimensional data, an effect known as the "curse of dimensionality" limits the applicability of density-based clustering (and other clustering paradigms). As distances are known to be more and more alike [7], it is no longer possible to meaningfully use the neighborhood notion for density assessment and cluster assignment. Intuitively, this means that irrelevant dimensions hinder the detection of clusters. Global dimensionality reduction techniques, such as Principal Components Analysis (PCA) may be able to reduce the data to a lower dimensional projection where clustering is possible. In many cases, however, the relevance of dimensions is not globally uniform, but varies locally. For some clusters, a certain subset of dimensions might be relevant, whereas for others, another subset is relevant. A single global reduction can therefore only detect some of the patterns, and loses those that are visible only in other projections.

To illustrate this effect in high dimensions, we give a toy example in two dimensions in Fig. 2. Assuming that we cannot cluster the two-dimensional full space directly, we can see that the projection to dimension D_1 results in a different set of subspace clusters than the projection to dimension D_2. Using only a single global projection, this cannot be uncovered. Please note that in high dimensional spaces, this effect is much more pronounced, and clusters visible in some subspace projections will display as random noise in other subspace projections.

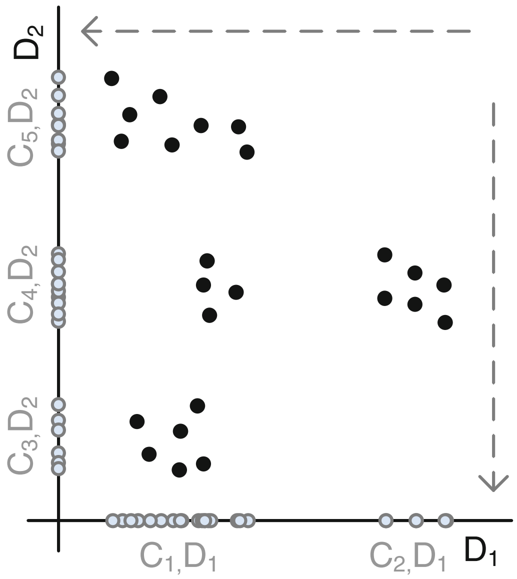


Fig. 2. Toy example of local relevance of dimensions: in projection to dimension D_1 different subspace clusters are discovered than in projection to D_2

In subspace clustering, the goal is the discovery of the relevant subspace projection for each cluster. As a consequence, the above definition of a density-based cluster can be easily adapted to subspace clustering by associating each cluster with its subspace, and by restricting the density-assessment in the neighborhood to this subspace.

Definition 2. ***Density-Based Subspace Cluster.*** *A density-based cluster C^S w.r.t. density threshold minPoints and neighborhood radius ε is defined as follows:*

– *All objects in C are dense, i.e., there are at least minPoints objects in their S-neighborhood:*

$$\forall o \in C : \ |N_\varepsilon^S(o)| = |\{p \in DB \mid dist^S(p,o) \leq \varepsilon\}| \geq minPoints$$

– *All objects are S-density-connected, i.e., for any two objects in the cluster, there is a chain of objects between them such that the successor in the chain is in the predecessor's neighborhood:*

$$\forall o,p \in C : \exists\ q_1,\ldots,q_m \in C : \forall_{i \in \{2,\ldots,m\}}\ q_i \in N_\varepsilon^S(q_{i-1})\ \wedge q_1 = o\ \wedge q_m = p$$

– *The cluster is maximal, i.e., any two density-connected objects are either both in cluster C or both not in it:*

$$\forall o, p \in DB: \; o, p \; S\text{-}density\text{-}connected \; \Rightarrow (o \in C \Leftrightarrow p \in C)$$

While this straightforward extension of DBSCAN to subspace clustering has been shown to successfully identify subspace clusters also in higher dimensional data [10], some specific challenges of high-dimensional data have been addressed. These challenges are two-fold: (i) regarding a model of density-based subspace clustering (dimensionality bias, redundancy) and (ii) regarding the efficiency of density-based subspace clustering (pruning, indexing), as discussed in the following sections.

3 Dimensionality Unbiased Density

When assessing the density around an object in full space, the neighborhood distance does not change. In subspace clustering, however, we restrict the density assessment to the subspace projection under consideration. While this is perfectly reasonable in order to identify density-connected groups in different subspace projections, a new issue arises when the overall clustering in all possible subspace projections is considered.

As the number of dimensions increases, the expected distance between any two data objects increases as well. As a consequence, the expected density drops accordingly. This means that comparing the number of data objects in the neighborhood of two subspaces of different dimensionality is biased towards the lower dimensional subspace. We say that the density assessment is dimensionality biased. This creates a problem when searching for all subspace clusters in all possible subspace projections. Either, one can set a low density threshold in order to ensure that also relatively high dimensional subspace clusters can be considered dense and thereby be discovered, and retrieve a huge number of low dimensional subspace clusters. Or, one can set a high density threshold to avoid overwhelming numbers of subspace clusters, and miss higher dimensional subspace clusters. Thus, dimensionality bias is a fundamental threat to meaningful subspace cluster discovery, as the distinction between dense clusters and sparse noisy areas is blurred by effects of the dimensionality.

In [3], a general definition of the notion of a dimensionality unbiased density measure is given:

Definition 3. ***Dimensionality Unbiased Density Measure****. A density measure $\varphi_\epsilon^{\mathbf{S}}$ is dimensionality unbiased if its expected density is the same for any two subspaces $\mathbf{S_1}$ and $\mathbf{S_2} \subseteq \mathbf{D}$:*

$$\forall \; \mathbf{S_1}, \mathbf{S_2}: \quad E\left[\varphi_\epsilon^{\mathbf{S_1}}\right] = E\left[\varphi_\epsilon^{\mathbf{S_2}}\right]$$

Using our notion above, the density measure $\varphi_\epsilon^{\mathbf{S}}$ is the count of objects within the neighborhood: $\varphi_\epsilon^{\mathbf{S}}(o) = |N_\varepsilon^S(o)|$. As discussed above, this density measure is

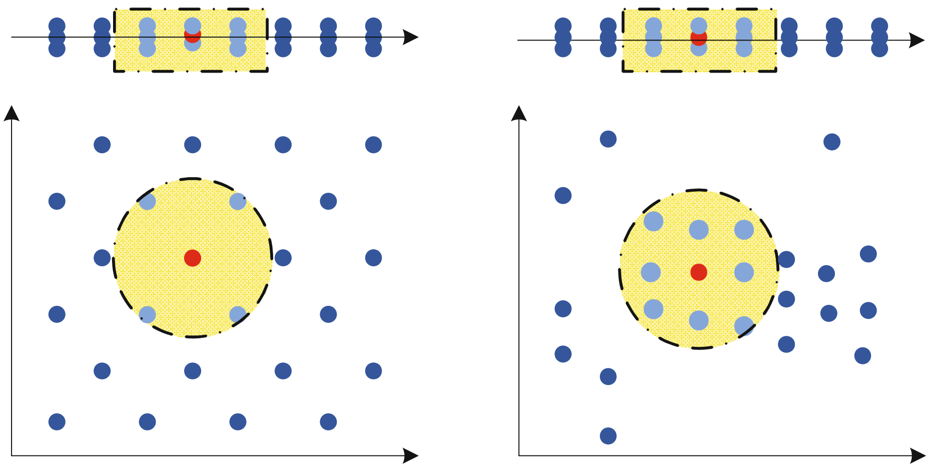


Fig. 3. Illustrating dimensionality bias: the expected density in a uniformly distributed space (left) in a one-dimensional space (top) is larger than in a two-dimensional space (bottom). Using a fixed density-threshold, a bias towards detection of low dimensional subspace clusters is observed. In the right figure, both neighborhoods contain the same number of points, which is more unexpected in a two-dimensional space.

biased, as increasing dimensionality leads to larger distances, and lower density within the neighborhood. An example is illustrated in Fig. 3. At the left, the situation in a uniformly distributed space is depicted for one-dimensional (top) and two-dimensional subspaces (bottom). As we can see, the expected number of objects within the chosen neighborhood size for this example is 9 and 5, respectively. At the right, we see an example in the two-dimensional space, where the neighborhood contains also 9 data objects, which is unexpected in this dimensionality, and should be recognized as the beginning of a subspace cluster. This can only be achieved by adapting the density measure such that bias is avoided.

Following [3], a density measure can be made unbiased by normalizing with its expected value:

Lemma 1. ***Eliminating Dimensionality Bias.*** *For any density measure* $\varphi_\epsilon^{\mathbf{S}}$

$$\frac{1}{E[\varphi_\epsilon^{\mathbf{S}}]}\varphi_\epsilon^{\mathbf{S}} \quad \textit{is dimensionality unbiased.}$$

Thus, by normalizing with the expected value within a subspace of a certain dimensionality, the density can be made unbiased. Going back to the example in Fig. 3, we can see that if we normalize with the expected value, then the situation at the bottom right is easily distinguished from the one at the top right. In this manner, it is easily possible to set the parameter for the density threshold such that both low dimensional and high dimensional subspace clusters can be detected. This is an important property for meaningful subspace clustering in high dimensional data.

4 Redundancy-Removal

Another issue that we are confronted with when moving from full space to subspace clustering, is the potential redundancy of the detected patterns. As noted before, a data object can be assigned to more than one cluster in different subspace projections. While this is a desirable property in order to uncover different patterns that might be able to describe diverse aspects of an object's alignment with different groups in the data, there is also a caveat. More precisely, a data object that is part of a subspace cluster of dimensionality k is likely to be also clustered similarly with other objects in dimensionality $k-1$, $k-2$, etc. Very often, these subspace clusters do not differ substantially, and therefore do not contribute any novel information for the user. By contrast, they can lead to overwhelming result sizes. At the most extreme, it may happen that more subspace clusters are reported than there are data objects in the entire database. Clearly, this is not an informative result.

We illustrate this in Fig. 4. As we can see, the two-dimensional subspace cluster is reflected in its two projections to both one-dimensional subspaces. Clearly, for higher dimensional subspace clusters, there are many more possible lower dimensional subspace projections. In the example, one of the reflected subspace clusters at the left includes a number of additional data objects, so it might be interesting to report this as a subspace cluster of its own, depending on user preferences. Please note, that we generally assume that among any two subspace clusters describing a given pattern, the higher dimensional one is the more interesting one, as it enumerates all dimensions in which we can describe the pattern. Additionally, of course, there can be subspace clusters that are not reflected in higher dimensional subspace clusters, such as the one-dimensional one at the lower left. This one should always be reported.

Therefore, research on subspace clustering has also investigated redundancy removal. In [5], redundancy is defined using a user preference parameter. This parameter specifies the relative amount of new data objects to be present in a non-redundant lower dimensional subspace cluster. In this manner, it is possible to trade-off result size with degree of new information.

Definition 4. ***Non-redundant Subspace Cluster.*** *A density-based cluster C^S is non-redundant w.r.t. redundancy parameter δ iff*

$$\neg\exists(C'^{S'})\textit{subspace cluster with } C' \subseteq C \wedge S' \supset S \wedge |C'| \geq \delta \ \cdot |C|$$

Interestingly enough, removing redundancy not only reduces the result to a manageable size, but also improves the overall quality of the result set. As demonstrated in the experimental evaluation in [5], the quality typically increases as redundancy is removed. This is largely attributed to the fact that redundant and smaller patterns often constitute noise or less prominent patterns that do not accurately grasp the overall structure of the data.

Recently, also more advanced definitions of redundancy have been studied, that consider the situation where a subspace cluster might be very similar to the union of two (or more) other clusters, and aim to optimize the result set

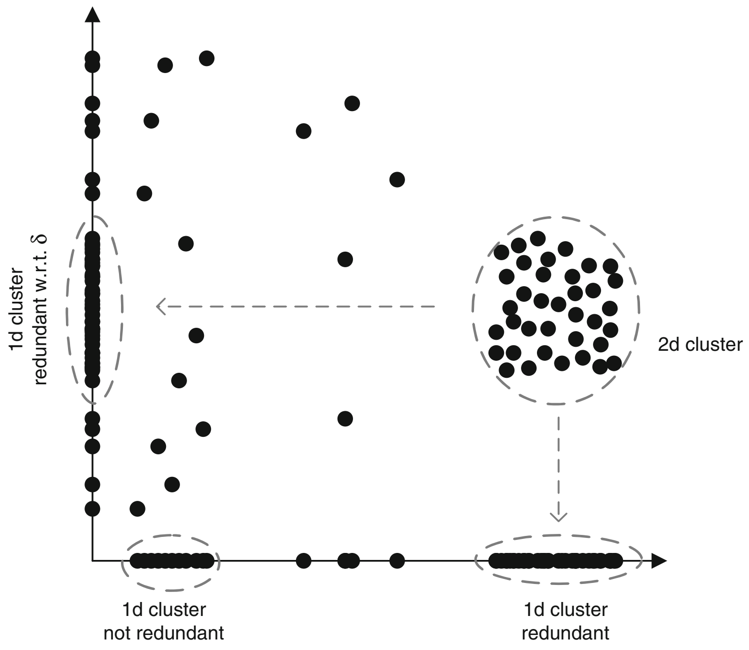


Fig. 4. Redundancy in subspace clustering: a two-dimensional subspace cluster is likely to be repeated similarly in its one-dimensional projections. If sufficiently many new data objects are included in the one-dimensional subspace cluster, it might be interesting to report as a separate pattern as well.

such that (almost) all data objects are described by a suitable subspace cluster [15]. Other approaches include [13], where the goal is to identify statistically significant patterns.

These models describe the desired output, i.e., which subspace clusters should be reported among the possibly large number of patterns in the different subspace projections. Naturally, the question arises how to efficiently detect all subspace clusters, and ideally only the non-redundant ones.

5 Pruning Subspace Clusters

Clearly, naively searching all possible subspace projections is not feasible for reasonably sized high dimensional data sets. To reduce the effort necessary in order to do density-based subspace clustering, especially for unbiased measures that are different for different dimensionalities, a multistep filter-and-refine algorithm has been introduced in [4]. An overview over the general idea is given in Fig. 5. As we can see, it is based on an initial conservative approximation of subspace cluster candidates using so called hypercubes. These hypercubes undergo a weak density filter that checks whether the candidate could be a subspace cluster in any dimensionality. Finally, if both filter steps are successfully passed,

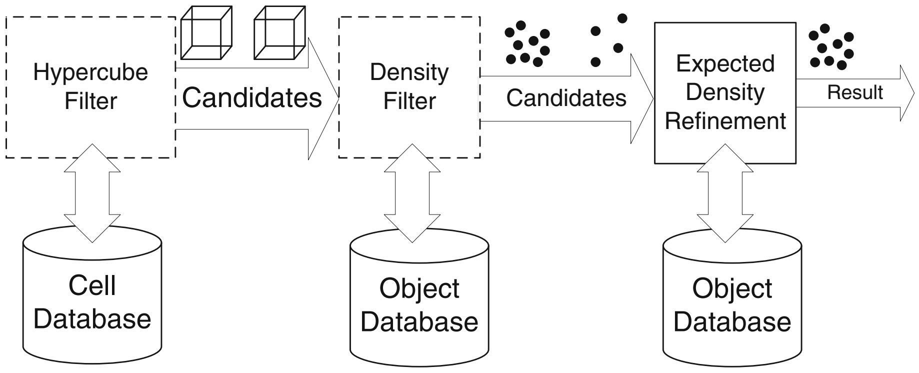


Fig. 5. Multistep filter-and-refine approach to subspace clustering: the data space is discretized losslessly via a density-conserving grid to generate hypercube candidates. Passing a weak density filter, only successful candidates are clustered with respect to the exact unbiased density measure in the corresponding subspace projection in order to generate the correct result.

the refine step subjects the candidate to the exact subspace clustering using the unbiased density measure. Since each filter step allows for pruning of sparse areas in space or weakly populated hypercubes, only few candidates typically undergo the expensive final clustering step. In this manner, subspace clustering is rendered considerably more efficient.

In the following, we discuss the filter steps in more detail. Initially, the data space has to be discretized in order to form a conservative approximation of potential subspace clusters in enclosing hypercubes. Please note that discretization is an approach that was also used in the first subspace clustering algorithm, CLIQUE [1]. Their grid discretization, however, may result in missed subspace clusters due to position or resolution of the grid. This issue is illustrated in Fig. 6. As we can see in the image to the left, the cluster in grid cells 2 and 3 in dimension 2 (grey area) is cut apart by the grid. Using a density threshold of e.g. five, this cluster would not be detected. By contrast, the sparsely populated area in cell 2 in both dimensions would pass the threshold and become part of the result.

In order to avoid this problem, the work in [4] uses a novel density-conserving grid that can detect such situations. As can be seen in the right part of the figure, additional borders at the top of the cell in each dimension are introduced. These borders are exactly the size of the neighborhood radius in density-based subspace clustering. Consequently, if there is no object in this border, no density-connection across the cell boundaries exist. Otherwise, if there is such an object, then there might well be a cluster that extends across more than one cell. Thus, the two cells would be merged to form what is called a hypercube. Processing all cells in a predefined order, all necessary merges are performed in the first hypercube filter step. At the end of this step, a conservative approximation of any possible subspace cluster has been generated. It can be described concisely

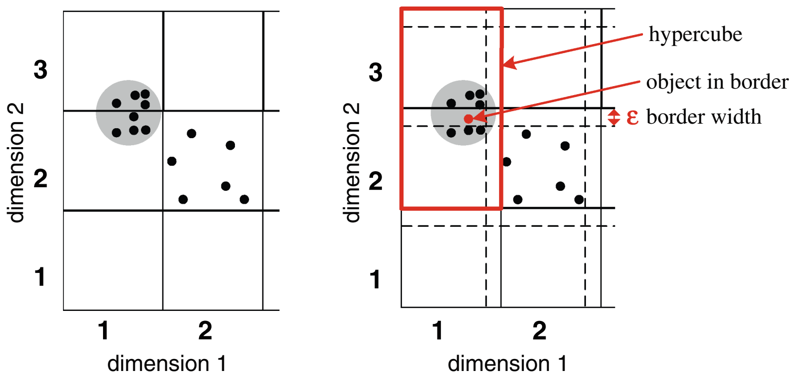


Fig. 6. Discretization approaches to subspace clustering: traditional grid (left) with issues regarding position and resolution, and the density-conserving grid (right) with additional density borders.

using the cell boundaries in the respective dimensions and the number of objects therein.

Once these hypercubes have been detected, a second filter step entails. It is based on the observation that any valid subspace cluster according to the unbiased density-based model has to exceed the density threshold for at least the weakest density measure. Since the expected density drops with the dimensionality, the weakest density measure is the one in the full space. Thus, the weak density filter simply checks the hypercube using this full space density assessment. Some hypercubes will be pruned in this step, while the remaining ones are subjected to the exact subspace clustering routine. After this refinement, all density-based subspace clusters have been detected. For details on how to prove this completeness, please see [4].

6 Indexing Subspace Clustering

The first subspace clustering algorithm CLIQUE uses an algorithmic approach similar in spirit to the apriori approach to frequent itemset mining [1]. The idea is to search the subspace projections in a bottom-up fashion. Starting from the one-dimensional projections, only subspace clusters from lower dimensions are combined to form candidates for higher dimensional projections. These candidates are verified, and if successful, used to generate candidates on the next higher level. This algorithmic approach is illustrated in Fig. 7 (left).

While this approach has been used successfully, also in density-based subspace clustering [10], its scalability is limited. This is due to the inherent principle used in bottom-up approaches: a high dimensional subspace cluster is reflected in its lower dimensional projections. Therefore, in order to generate the interesting high dimensional subspace clusters, a very large number of (redundant) lower dimensional subspace clusters has to be generated first. Also, these methods

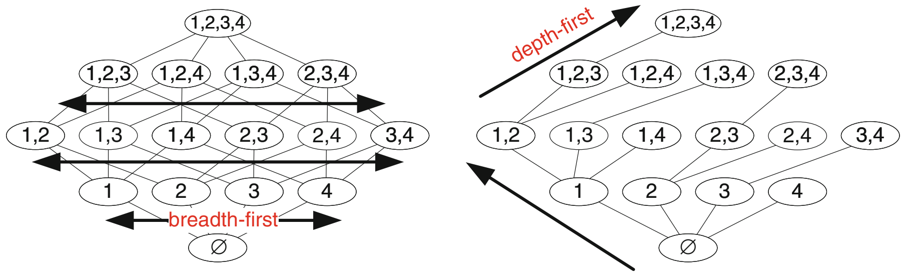


Fig. 7. Left: bottom-up or breadth-first approach to subspace clustering: starting from one-dimensional projections, subspace clusters are used to generate candidates on the next higher level. Entire levels have to be generated before moving to the next higher level. Right: depth-first approach to subspace clustering: identify the highest possible subspace projection that contains subspace cluster candidates, and prune any lower dimensional redundant subspace cluster if successful.

assume a monotonicity property of the density measure. For unbiased density measures, they therefore do not work.

As an alternative, a depth-first approach based on the above discretization has been proposed in [5]. As a key advantage, it proposes an index structure that is capable of handling all different subspace projections. For bottom-up approaches such an index structure cannot be devised, as all possible combinations of dimensions would need to be available in the index.

The idea of the so-called SCY-tree is to use the discretized representation discussed above to create a compact representation of the data. Each cell count is mapped to a node in the tree, including border counts. An example is given in Fig. 8: at the left, we see the density-conserving grid representation in two dimensions. To the right, the corresponding SCY-tree in three dimensions is depicted. The highlighted area to the left of the grid corresponds to all three data objects that are located in cell one in dimension one. This is reflected in the grey node "1:3" at the leaf level which records three objects in cell one (note that the leaf level corresponds to dimension one in the example). The highlighted border object at the top of cell three connects cells one and two in dimension two. Correspondingly, there is a special node "1" at the middle level in the tree between the nodes that contain the cell counts for cell one and cell two. The highlighted path from the root to the leaf tells us that there are five objects located in cell two in dimension three (not visible in the grid), out of which four are located in cell two in dimension two, out of which three are located in cell one in dimension one. In this manner, it is possible to identify the location of all objects in all dimensions.

The SCY-tree contains all the information necessary to analyze the counts in all subspace projections without having to go back to the database. Algorithmically, this is similar in spirit to the FP-tree mining algorithm for frequent itemset mining [9]. The idea is to cut out the relevant paths to form so-called restricted trees that contain only counts in the respective projections. This is

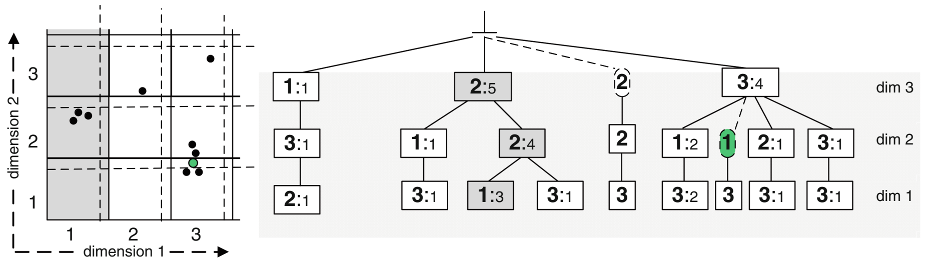


Fig. 8. SCY-tree index structure for subspace clustering: nodes record the cell id and the cell count, the level in the tree indicates the dimension. A path from the root to a leaf corresponds to the exact count in a full space cell.

illustrated in Fig. 9: to obtain a restricted tree for a particular cell or dimension, the corresponding nodes on that level are identified and the paths for the relevant subspace projections are extracted. Please note that the merge of several paths or trees is straightforward: simply sum up the corresponding counts.

The flexibility in this extraction of information in arbitrary subspace projections makes it possible to work in a depth-first manner (see also Fig. 7 (right)). The core idea is to build hypercubes of the highest possible dimension, i.e., such that the count or density thresholds are still met. Then, these candidates are subjected to the filter steps outlined above. Only if these filter criteria are not met, the algorithm will consider a lower dimensional projection. Otherwise, if all filters are successfully passed and the refinement step discovers a valid subspace cluster, then all redundant lower dimensional projections can be immediately discarded without even being generated. This is a major advantage over bottom-up approaches that have to work their way up through all redundant candidates in order to reach the more interesting higher dimensional projections.

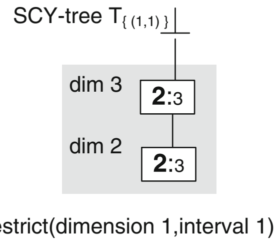


Fig. 9. Working directly on the index structure: extracting count information for a particular subspace projection is straightforward. In this example, identify all nodes corresponding to cell one in dimension one (leaf level) to obtain a restricted SCY-tree that contains the corresponding counts in dimensions two and three.

7 Approximate Jump Clustering

In extremely large scale or very high dimensional data sets, the run times of subspace clustering remain an issue. In order to further improve the efficiency of subspace clustering algorithms, approximate approaches have been proposed. By relaxing the completeness constraint, the idea is to identify promising regions in higher dimensional subspace projections just from a few lower dimensional indicators [16]. Figure 10 illustrates such a jump from several one-dimensional to a single three-dimensional region. Please note that in general, several jumps to higher dimensional subspace projections are typically used. This type of processing can be likened to traditional best-first search: the processing order of subspace projections is based on priorities using information on promising regions from lower dimensional subspace projections.

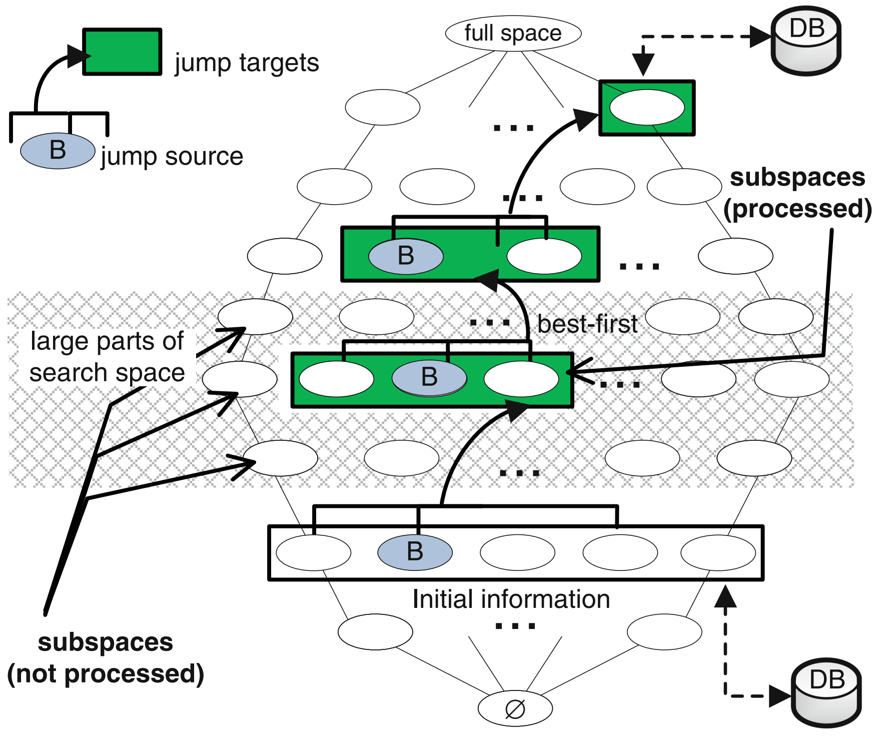


Fig. 10. Approximate "jump" subspace clustering or best-first search: based on initial information in low dimensional subspace projections, few promising regions in higher dimensional subspace projections are selected for a direct jump that avoids intermediate subspace projections all together.

The general idea is to maintain a hierarchy of information at different levels of granularity that allow the subspace clustering algorithm to steer through the search space of regions in different subspace projections. All information obtained by the algorithm is organized in priority queues. As illustrated in Fig. 11, the lowest level contains initial density estimates (one- or two-dimensional histograms) for all regions. From these, hyperrectangles are formed at the next higher level, then actual subspace clusters. The entire process is steered such that preference is given to the most promising regions that have not yet been covered.

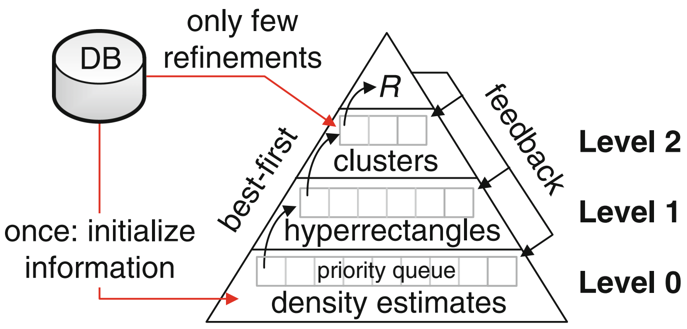


Fig. 11. Hierarchy of information maintained in priority queues for subspace clustering: at the lowest level, density estimates (histograms) initialize the process, at level 1, hyperrectangles are stored, and level 2 maintains actual subspace clusters.

Please note that we can once again make use of the issue of redundancy in order to obtain a good steering strategy: we know that the clusters in subspaces of high dimensionality are likely to be reflected in very many low dimensional subspace projections. This effect is put to good use by giving priority to hyperrectangles that are very similar and therefore likely to be reflections of the same subspace cluster. On the other hand, priority is given to regions that include data objects that are not yet covered, i.e., that are not yet assigned to any subspace cluster. In this manner, the subspace clustering aims to find an (almost) complete subspace clustering of the data objects that avoids mining large areas of the search space. Experimentally, not only substantial efficiency gains, but also good quality results are found [16].

8 Conclusion

Density-based clustering is a successful approach to automatically grouping data based on mutual similarity even in noisy data. In high-dimensional data, traditional clustering approaches fail due to effects attributed to the "curse of dimensionality". This paper discusses work done on density-based subspace clustering in such high-dimensional data.

In subspace clustering, the goal is to find clusters in the subspace projections that are most relevant to describe them. As discussed in this piece of work, a number of specific challenges arise for density-based subspace clustering. Regarding suitable models of subspace clustering, removing dimensionality bias is an important issue. As the dimensionality increases, the density assessment should still be capable of reliable differentiating dense areas from noise. Another concern is the pruning of redundant subspace clusters that contain more or less the same information.

From an efficiency point of view, this is also helpful in that non-interesting lower dimensional subspace clusters do not need to be generated. A multistep filter-and-refine approach can further reduce run times by identifying conservative approximations based on density-conserving grids. This approach is supported by an index structure that maintains cell counts for tree-based depth-first processing. For most efficient processing, the completeness constraint can be relaxed in favor of an approximate jump subspace clustering technique that maintains information on promising regions to be processed in a best-first fashion.

The above advances in density-based subspace clustering still face limitations for extremely high dimensional data as is common e.g. in bioinformatics. For these application domains, further efficiency improvements are necessary. At the same time, there are often different similarity measures that are usually not studied in density-based subspace clustering. The recent interest in the data mining community in parallel computation techniques such as Hadoop opens for exciting new research that will make it possible to make full use of modern hardware.

Acknowledgments. This work has been supported in part by the Danish Council for Strategic Research, grant 10-092316, and by the Danish Council for Independent Research - Technology and Production Sciences (FTP), grant 10-081972.

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: SIGMOD, pp. 94–105 (1998)
2. Assent, I.: Clustering high dimensional data. Wiley Interdisc. Rev. Data Min. Knowl. Discov. **2**(4), 340–350 (2012)
3. Assent, I., Krieger, R., Müller, E., Seidl, T.: DUSC: dimensionality unbiased subspace clustering. In: ICDM, pp. 409–414 (2007)
4. Assent, I., Krieger, R., Müller, E., Seidl, T.: EDSC: efficient density-based subspace clustering. In: CIKM, pp. 1093–1102 (2008)
5. Assent, I., Krieger, R., Müller, E., Seidl, T.: INSCY: indexing subspace clusters with in-process-removal of redundancy. In: ICDM, pp. 719–724 (2008)
6. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (1957)
7. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is nearest neighbor meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
8. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases. In: KDD, pp. 226–231 (1996)

9. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD, pp. 1–12 (2000)
10. Kailing, K., Kriegel, H.-P., Kröger, P.: Density-connected subspace clustering for high-dimensional data. In: SDM, pp. 246–257 (2004)
11. Kriegel, H.-P., Kröger, P., Sander, J., Zimek, A.: Density-based clustering. Wiley Interdisc. Rev. Data Min. Knowl. Disc. **1**(3), 231–240 (2011)
12. Kriegel, H.-P., Kröger, P., Zimek, A.: Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans. Knowl. Discov. Data **3**(1), 1:1–1:58 (2009)
13. Moise, G., Sander, J.: Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In: KDD, pp. 533–541 (2008)
14. Moise, G., Zimek, A., Kröger, P., Kriegel, H.-P., Sander, J.: Subspace and projected clustering: experimental evaluation and analysis. Knowl. Inf. Syst. **21**, 299–326 (2009). doi:10.1007/s10115-009-0226-y
15. Müller, E., Assent, I., Günnemann, S., Krieger, R., Seidl, T.: Relevant subspace clustering: mining the most interesting non-redundant concepts in high dimensional data. In: ICDM, pp. 377–386 (2009)
16. Müller, E., Assent, I., Günnemann, S., Seidl, T.: Scalable density-based subspace clustering. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 1077–1086. ACM (2011)
17. Müller, E., Günnemann, S., Assent, I., Seidl, T.: Evaluating clustering in subspace projections of high dimensional data. PVLDB **2**(1), 1270–1281 (2009)
18. Parsons, L., Haque, E., Liu, H.: Subspace clustering for high dimensional data: a review. SIGKDD Explor. **6**(1), 90–105 (2004)
19. Sim, K., Gopalkrishnan, V., Zimek, A., Cong, G.: A survey on enhanced subspace clustering. Data Min. Knowl. Discov. **26**, 332–397 (2012). online first

Comparing Fuzzy Clusterings in High Dimensionality

Stefano Rovetta[1](✉) and Francesco Masulli[1,2]

[1] DIBRIS – Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi, Università di Genova, Genova, Italy
stefano.rovetta@unige.it

[2] Center for Biotechnology, Temple University, Philadelphia, USA

Abstract. Due to the specificity of clustering, a problem that is intrinsically ill-posed, there are several approaches to comparing clusterings. Comparison of clusterings obtained in different conditions is often the only affordable evaluation strategy, due to the lack of a ground truth. In this chapter we address a class of dimensionality-independent methods which can be applied in the presence of a high-dimensional input space. Specifically, we review some generalizations of this class of methods to the case of fuzzy clustering, in several variants.

1 Introduction

High-dimensional data is encountered in most fields of science and technology. Although progress in data analysis and processing methods constantly changes the concept of what constitutes high dimensionality, there are some aspects of the problem which are inherent and unescapable, since they are more related to the ratio between data dimensionality and cardinality than to absolute values of either.

The most well-known description of such phenomena is termed the curse of dimensionality, which expresses the consequences of volume growing exponentially with the number of dimensions. These consequences include for instance:

– Sparsification of data (the *empty space* phenomenon): In many cases the number of observations (cardinality) is comparable with or even lower than the number of observed variables (dimensionality).
– Exponentially growing number of model parameters, with corresponding growth of necessary observations to obtain a given level of confidence or precision.
– Concentration effect on distances: For metrics of a very general form, maximum and minimum observed values tend to take on the same value with a probability that grows very rapidly with dimensionality; therefore distances are not meaningful any more.

While supervised analysis (for instance, classification) can count on a rich set of methods to keep the effects of dimensionality under control, such as model

F. Masulli et al. (Eds.): CHDD 2012, LNCS 7627, pp. 50–71, 2015.
DOI: 10.1007/978-3-662-48577-4_4

complexity estimation and working with kernels, for unsupervised methods and especially for clustering the same tools are not always available. This is because, while classification works with the mapping from data to nominal labels, clustering works directly with the structure of the space where the data live. Nonetheless, several techniques are commonly encountered for enabling clustering in high dimensions. These include (see also [29] in this book):

- Variable selection: Retain only the variables that are deemed significant to the problem at hand, according to some criterion.
- Subspace clustering: Perform clustering in spaces defined by a subset of the variables, and then search for the most meaningful subset. Alternatively, find a subspace (a more general linear projection, not necessarily axis-parallel) where clustering is most satisfactory.
- Intrinsic dimensionality estimation followed by (possibly nonlinear) dimensionality reduction: Find the dimension of the subset of the data space where the data actually "live", which most often is much lower than the number of observed variables, and then map data onto a lower-dimensional structure, which can be linear (a subspace), locally linear (union of several subspaces, each restricted to a given region), or non-linear (a manifold).
- Specialized metrics: Find ways to measure (dis)similarity that are less affected by concentration effects, for instance with particular values of the exponent in Minkowski metrics, or by using ranks instead of primary measures.
- Working with an affinity matrix rather than directly with data, using specific methods that do not require the direct representation of objects: Agglomerative clustering, correlation clustering, shared neighbors clustering.
- Using kernel and spectral clustering, which start from data representations and map them into affinity-based representations by using specific measures (kernels).

In this contribution we describe some techniques to measure similarities between pairs of different clusterings, taking advantage of the added flexibility provided by fuzzy clustering. We will review a few existing clustering similarity indexes, and describe some possible generalisations and extensions that make them applicable even in the fuzzy case. Some applications, using benchmark data sets, will also be shown.

A recent extensive survey [12] cites 76 measures of similarity or dissimilarity developed over the last century. The same problem can be cast as measuring diversity among classifiers or clusterings, binary string similarity, categorical feature similarity. A more recent trend has been to incorporate more information than just the coincidence of binary/categorical attributes; this includes for instance the development of fuzzy variants [10, 40].

2 Fuzzy Clustering

2.1 Some Notations and Definitions

The task of data clustering can be defined using set-theoretic concepts. A *clustering* of a given data sample, a set of N data points in a metric space

$X = \{\mathbf{x}_1, \dots \mathbf{x}_N\}$, can be defined as a partition of the sample itself. This identifies *partitional* clustering methods. So clustering seeks a K-partition $\Pi = \{C_1, \dots, C_K\}$ of X. Each "part" of the partition is a cluster C_j, represented by a centroid from a set $Y = \{\mathbf{y}_1, \dots, \mathbf{y}_K\}$. Note that these definitions allow us to extend partitions from the finite sample at hand to the whole data space where the data "live", thus providing what is commonly called an "out-of-sample extension".

It is customary to express attribution of data point $\mathbf{x}_l$ to cluster C_j by means of an indicator function $u(\mathbf{x}_l, C_j)$, the *membership function.* In the presence of a finite data set X, the membership values are the entries $u_{lj} = u(\mathbf{x}_l, C_j)$ of the *membership matrix* U, whose rows are *membership vectors* $\mathbf{u}_l$.

2.2 Fuzzy Clustering

When we want to take into account more refined models, with more information made available through the representation of clusters, we can resort to a *fuzzy* formalism [45]. In general, there are several ways to represent uncertainty with fuzzy sets, but in the special case of fuzzy data clustering [42] "fuzzy" means specifically representing partitions by means of real-valued indicator functions. This implies that a fuzzy clustering is not a conventional partition, but rather a fuzzy partition allowing for partial overlap of clusters.

When compared to standard clustering, fuzzy clustering provides a more flexible and powerful data representation paradigm. Fuzzy partitional methods based on centroids share some model parameters with their closest non-fuzzy counterparts, the number of clusters being the most notable example. However, most of them also require setting some additional parameters, which often play the role of degrees of fuzziness. As basic examples, we can mention Bezdek's fuzzy c-means [7] which needs the exponent m to be set to control fuzziness, and Krishnapuram and Keller's possibilistic c-means [26] which requires a set of width parameters β_j, one per cluster. In [30], we have proposed a *graded possibilistic c-means* clustering technique (GPCM) that provides control over the degree of possibility, thus allowing a soft transition between the standard probabilistic and the possibilistic models. This is done through an additional parameter α.

2.3 Methods for Fuzzy Clustering

Now we briefly review some fuzzy clustering methods, characterized by the fact that cluster centroids are defined as follows:

$$\mathbf{y}_j = \frac{\sum_{l=1}^{N} u_{lj} \mathbf{x}_l}{\sum_{j=1}^{K} u_{lj}}. \tag{1}$$

This formulation characterizes all the methods derived from c-means and is obtained from the minimization of a suitable Lagrangian, but does not depend on the actual computation of the memberships.

However, the membership function will differ according to the specific method; therefore the resulting centroids, whose computation depends on memberships as per Eq. (1), in general will not be the same in different cases.

One taxonomy of methods uses the constraint imposed on the sum of all membership for any given data point $\mathbf{x}_l$,

$$\zeta_l = \sum_{j=1}^{K} u_{lj}, \tag{2}$$

as a discriminant feature. It is useful to think of a membership vector $\mathbf{u}_l$ as a point in the K-dimensional space of possible combinations of memberships for the given point. The different feasible sets of membership values characterise each specific method, as detailed in the following.

When the sum of memberships is constrained to $\zeta_l = 1$, we are in the standard "probabilistic" case. With the usual formulation for crisp clustering, where $u_x(C) \in \{0, 1\}$, subject to the sum-1 constraint, only a set of K possible configurations is available, namely, those corresponding to the membership vectors that lie on the coordinate axes, a subset of the vertices of the unitary K-hypercube. Here one and only one of the memberships can be 1, while all others are zero.

A more interesting and expressive case is that of fuzzy clustering. Here the memberships lie on a segment of the K-dimensional hyperplane

$$\zeta_l = 1. \tag{3}$$

More specifically, they are located on the diagonal of the K-hypercube $[0, 1]^K \in \mathbb{R}^K$. Memberships obeying this constraint are formally equivalent to probabilities. This case is termed "probabilistic" to stress this analogy. Crisp clustering is a limit case of general probabilistic clustering, where "probabilities" correspond to certainty; crisp memberships can only be located at the vertices of the hypercube.

The Maximum Entropy (ME) approach [38,39] makes explicit use of the probabilistic interpretation of memberships. In ME, by imposing the necessary minimum condition on an objective function with an entropic penalty, the problem can be stated as a minimization of the following Lagrangian:

$$J_{\mathrm{ME}} = \sum_{l=1}^{N} \sum_{j=1}^{K} \left[u_{lj} d_{lj}^2 + \eta u_{lj} \log u_{lj} \right], \tag{4}$$

and as a result of computing the necessary minimum conditions, memberships can be obtained from:

$$u_{lj} = \frac{e^{-d_{lj}/\beta}}{Z_l}. \tag{5}$$

where $Z_l = \sum_{j=1}^{K} e^{-d_{lj}/\beta}$ is termed the partition function. Clusters C_j then obey the Gibbs distribution around the respective centroids $\mathbf{y}_j$:

$$\Pr\left(\mathbf{x} \mid \Pi_j\right) = \frac{e^{-\beta \|\mathbf{x}-\mathbf{y}_j\|^2}}{Z(\mathbf{x})}, \tag{6}$$

where $\mathbf{y}_j$ is the centroid representing cluster $\mathbb{C}_j$ and β is a resolution parameter that, in a thermodynamic analogy, plays the role of a temperature. The quadratic distortion is the "energy" of "particle" $\mathbf{x}_l$ and $Z(\mathbf{x}_l)$ is the corresponding "partition function" at the specific "temperature" value $1/\beta$:

$$Z(\mathbf{x}) = \sum_k e^{-\beta||\mathbf{x}_l - \mathbf{y}_k||^2}. \tag{7}$$

The optimization procedure proposed for this model includes an "annealing" schedule to gradually lower the system's temperature. Since at each step a stable state is reached before moving to the next step, this method is also called Deterministic Annealing.

2.4 Possibilistic Clustering Models

The Possibilistic c-Means (PCM) [3,25,26] can be seen as being located at the other end of the spectrum with respect to the probabilistic Maximum Entropy method. It is based on removing any equality constraint on the sum of memberships, replaced by a set of loose requirements, which essentially allow the memberships themselves to take any configuration within the hypercube $[0,1]^K$, with the exception of two isolated points, those with all-zero and all-one memberships, respectively. These are excluded by design by means of additional checks to avoid trivial solutions. Note that now the memberships are not formally equivalent to probabilities any more.

In this *possibilistic* case, taking as a reference the second formulation presented in [26], the objective function has the form

$$J_{\mathrm{PCM}} = \sum_{l=1}^{N} \sum_{j=1}^{K} \left[u_{lj} d_{lj}^2 + \eta_j \left(u_{lj} \log u_{lj} - u_{lj} \right) \right], \tag{8}$$

and, again per the necessary minimum conditions, memberships are computed as

$$u_{lj} = e^{-d_{lj}/\beta_j}. \tag{9}$$

If we set a single value β for all the β_j, the only difference with Eq. 5 is in the denominator. To take advantage of this fact, we generalize the membership function as follows:

$$u_{lj} = \frac{v_{lj}}{Z_l}, \tag{10}$$

where we have introduced the *free membership* v_{lj}, defined as follows:

$$v_{lj} = e^{-d_{lj}/\beta_j}. \tag{11}$$

These functions share the same term for penalizing the overall distortion, but each of them has different additional penalties. As a result, the centroid location update equations remain the same, resulting in centers being placed at the barycenter of clusters weighted by membership. The membership update equations, which as per Eqs. 5 and 9 express the dependence of memberships from distances, differ by the form of the term Z_l. For a general class of clustering formulations and associated objective functions, we may redefine ζ_l in terms of the free memberships

$$\zeta_l = \sum_{j=1}^{K} v_{lj}. \tag{12}$$

For instance, in the probabilistic approaches $Z_l = \zeta_l$, whereas in standard possibilistic approaches $Z_l = 1$.

2.5 Graded Possibilistic Models

The classic probabilistic membership model, be it either hard or fuzzy, implements the concept of partitioning a set into disjoint subsets with memberships formally equivalent to the probability of one out of K mutually exclusive events. In the possibilistic approach each membership is formally equivalent to the probability of one out of K mutually *independent* events. Of course they may retain the usual fuzzy interpretation as degrees of truth rather than probabilities.

The graded possibilistic membership model assumes instead that events may be independent to a certain degree, but not completely, so that, while intermediate cases will be treated as independent, extreme cases (with some very high or very low membership values) will be considered mutually exclusive. This provides the method with a notable expressive power in terms of fuzzy modelling.

The partition function characterizing the Graded Possibilistic c-Means (GPCM) is derived from the *interval* constraint $\sum_{j=1}^{K} {u_{lj}}^{[\gamma]} = 1$. Here we use an interval variable $[\gamma] = [\,\gamma^{(l)}, \gamma^{(u)}\,]$. Note that ${u_{lj}}^{[\gamma]} = [\,{u_{lj}}^{\gamma^{(l)}}, {u_{lj}}^{\gamma^{(u)}}\,]$ since an exponential with interval exponent $[A] = [\underline{A}, \overline{A}]$ is the interval $e^{[A]} = [e^{\underline{A}}, e^{\overline{A}}]$ [34].

An interval variable is commonly interpreted as the *admissible range* for the actual value of an unknown variable. Adopting this interpretation, the mixed-type equality between a non-interval variable a and an interval variable $[A]$ has been conventionally used with the following meaning: The equality $a = [A]$ is true when $\underline{A} \le a \le \overline{A}$, or $a \in [A]$. In most applications of interval arithmetic, from numerical error bracketing to type-2 fuzzy sets, this means that $[A]$ is the uncertain representation of a.

In Ref. [30] this is explained in some more detail; here we restrict ourselves to a particular choice of $\gamma^{(l)}$ and $\gamma^{(u)}$, for which we obtain the specific implementation that we study in this work: $\gamma^{(l)} = \alpha$ and $\gamma^{(u)} = 1$, where $\alpha \in (0, 1]$ controls the "possibility level." In other words, the interval parameter $[\gamma]$ has the form

$$[\gamma] = [\alpha, 1]. \tag{13}$$

In this specific, asymmetric implementation, memberships whose sum exceeds 1 are forbidden. Therefore clustering is effectively competitive among nearby centroids. However, for far-away centroids, the competition decreases with α. This allows us to obtain points which are not attributed to any cluster, thus providing a very natural representation for outliers. The partition function is in this case computed as follows:

$$\begin{cases} Z_l = \sum_{j=1}^{K} v_{lj} & \text{if} \quad \sum_{j=1}^{K} v_{lj} > 1 \\ Z_l = \left(\sum_{j=1}^{K} v_{lj}^{\alpha}\right)^{1/\alpha} & \text{if} \quad \sum_{j=1}^{K} v_{lj}^{\alpha} < 1 \\ Z_l = 1 & \text{otherwise.} \end{cases} \tag{14}$$

For $\alpha = 1$, the representation properties of the method reduce to those of ME, while in the limit case for $\alpha \to 0$, the representation properties are equivalent to those of PCM-II for low membership values, and to those of ME for higher values.

This method, the Asymmetric Graded Possibilistic c-Means (AGPCM), has several nice properties that make it worth studying and using in practice:

- In [30] this particular model has been shown to possess robustness properties. The rejection ability can be applied in robust clustering and outlier analysis, while its reverse, outlier identification, can be used in novelty detection and data distribution characterization (or one-cluster clustering) as in [15].
- While for the fully possibilistic method it is very difficult to attain convergence, the graded approach has better convergence for α sufficiently larger than 0. It is also possible to "play" with parameters along the optimization process, for instance applying an annealing schedule to α, so as to exploit the best values in the most appropriate phase of the convergence: higher α in the initial steps, when centroids need to break symmetry and diverge; lower α in the later steps, when a precise, outlier-insensitive placement is sought.
- From the point of view of data analysis, full membership to more than one cluster, as allowed by a symmetric formulation, may have a difficult interpretation; in contrast, a point which does not belong to any cluster is easily interpreted as an outlier.
- As a quantitative counterpart of the previous point, memberships summing up to *at most* one allow a much easier comparison between clusterings, since the range of values for fuzzy similarity indexes depends on the values of memberships. This point will be discussed further on in this paper.
- Outlier insensitivity presents advantages with respect to convergence as well, since, while centroids in non-fuzzy clustering are insensitive to points outside their cluster, centroids in fuzzy clustering have to account for the effect of all points. This is not the case with the possibilistic model. An illustration of this increased precision in locating cluster centres is provided in Fig. 1.
- On the other hand, with respect to PCM and symmetric-GPCM, AGPCM features an effective repulsion between nearby centroids, thus reducing the risk of overlapping clusters.

The main disadvantage of this method is the presence of relatively many parameters that need to be set. No criterion was given in the original work to

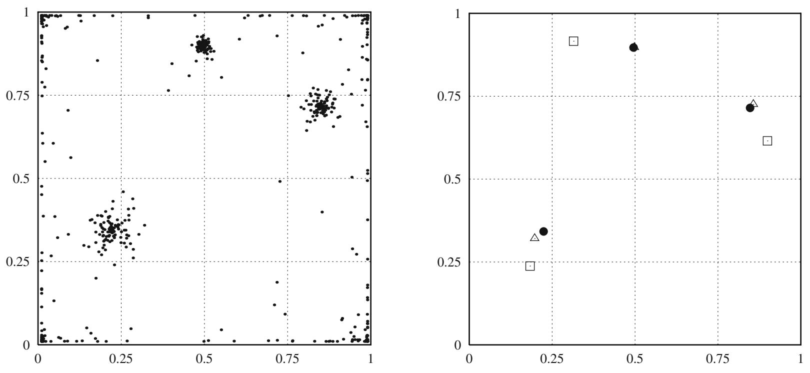


Fig. 1. Outlier rejection demonstration for AGPCM. Above: Data set. Below: Centroid locations. Black circles are the true cluster centres; triangles are centres found with $\alpha = 0$ (maximum rejection); squares are centres found with $\alpha = 1$ (no rejection, representationally equivalent to ME). Note that some triangles are hidden by true cluster centres since they are almost coincident; this is clearly not true for the "probabilistic" centroids. Figures from [30].

assess the value of α, while β was initialized following heuristics available in the literature; for both, "annealing" procedures were then applied, but without any measure of quality. The stability analysis described in the following sections was motivated by this lack of objective tools.

3 Comparing Fuzzy Clusterings

3.1 Approaches to the Comparison of Clusterings

Measuring the agreement between two clusterings amounts to measuring the similarity between two partitions, and there are several partition similarities available in the literature. It should be noted however that fuzzy clustering is not addressed very frequently in these works, even if it has several advantages over standard clustering from both representational and computational viewpoints [24,30]. The two main approaches include comparing clusters after matching them, and comparing co-association information.

The first approach consists in first identifying pairs of clusters, each composed of one cluster from the first partition and one from the second, which can be considered related, or, ideally the same cluster. A perfect match is difficult to obtain, and this *correspondence problem* may not have a satisfactory solution. The second step is to evaluate the degree of matching, and this is of course possible only if the first step succeeded.

A second approach is based on co-association. Two data items are co-associated if a partition puts them in the same cluster. The agreement or disagreement of partitions can be measured by coassociation, i.e., counting the number of pairs of data items on which both partitions agree, and comparing it with the number of pairs on which they disagree. Several classic indexes are based on this rationale. In the following we review some of them and describe some variations, including fuzzy and probabilistic versions.

This approach is followed in [16,21]. In [40] we defined a methodology to extend several indexes based on co-association to the fuzzy and possibilistic case, by means of co-association matrices [17]. Several indexes can simply be computed starting from the entries of a confusion or contingency matrix, which is readily obtained from the co-association matrix.

3.2 Notation

Suppose we select a partitional clustering method. For a specific choice of clustering parameters, possibly including selecting a given number of clusters and a random initialization, but excluding changes in the data sample, we obtain a given partition. If we repeat this process for a number of times, the i-th run will optimize a set $Y^i = \{\mathbf{y}_1^i, \ldots, \mathbf{y}_{K^i}^i\}$ of K^i centroids, which represent a K^i-partition $\Pi^i = \left\{C_1^i, \ldots, C_K^i\right\}$. We will not require that $K^i = K^k$ for $i \neq k$, i.e., it is not necessary to have the same number of clusters in different instances.

The i-th fuzzy indicator function will be similarly denoted as $u^i(\mathbf{x}_l, C_j^i)$, and likewise we will have $u_{lj}^i = u^i(\mathbf{x}_l, C_j^i)$. However, wherever we do not refer to specific instances nor need to differentiate among them explicitly, we will drop the sub/superscripts i, k to avoid a cumbersome notation.

We indicate the fact that partition Π^i puts two data items x_l and x_m in the same cluster by writing the indicator function $x_l \sim_i x_m$. The negation is expressed with the barred symbol: $x_l \nsim_i x_m$. This notation is borrowed from [5].

3.3 Co-association

In fuzzy clustering partitions are fuzzy, meaning that $\forall \mathbf{x}_l \in Xl : 1, \ldots, N$, the membership $u_{lj} = u(\mathbf{x}_l, C_j) \in [0, 1]$ for each cluster $C_j \in \mathcal{P}$ and $\forall l : 1, \ldots, N$ the constraint $\sum_{j=1}^K u_{lj} = 1$ holds as per Eq. (3); in addition, we allow possibilistic partitions by removing this last constraint. As discussed earlier, possibilistic partitions may be a meaningful extension to fuzzy partitions especially in the asymmetric constraint case (where $\sum_j u(\mathbf{x}_l, a_j) \leq 1$), although they can also be considered in symmetric graded cases and in fully possibilistic cases with a bit more interpretation effort.

Under a given partition Π, each data point is now represented by a membership vector. We define the *coassociation* ξ_{lm} between two data items $\mathbf{x}_l$ and $\mathbf{x}_m$ as the degree of similarity between the representation of the two items under the partition Π. Extending the notation of [5], we compute $\xi_{lm} = \mathbf{x}_l \sim \mathbf{x}_m$ as follows:

$$\xi_{lm} = \sum_{j=1}^{K} u_{lj} \wedge u_{mj}. \tag{15}$$

We can also define the *negative coassociation*, which is the logical complement of the coassociation:

$$\mathbf{x}_l \nsim \mathbf{x}_m = \overline{\xi_{lm}}. \tag{16}$$

Note that in the non-fuzzy case these definitions collapse to the propositions "partition Π puts/does not put $\mathbf{x}_l$ and $\mathbf{x}_m$ in the same cluster", but in the fuzzy case it is necessary to take all clusters into considerations because, in general, none of them will be exactly zero or one. On the other hand, in the fuzzy case, ξ_{lm} is a *degree* of coassociation rather than an integer value representing binary logic conditions.

3.4 Fuzzy Coassociation

To obtain a specific fuzzy instantiation of the general definition of coassociation just given, we have to appropriately define the conjunction connective [45]. We adopt the product t-norm [33], which provides uniformity with respect to other models of imprecision or uncertainty. The conjunction logical connective under the product t-norm is defined as $a \wedge b = ab$, and the negation operator as $\overline{a} = 1 - a$, so that $a \vee b = a + b - ab$ (the *probabilistic sum* t-conorm). The fuzzy coassociation between $\mathbf{x}_l$ and $\mathbf{x}_m$, therefore, is a real value ξ_{lm} computed as

$$\mathbf{x}_l \sim \mathbf{x}_m = \xi_{lm} = \sum_{j=1}^{K} u_{lj} u_{mj} = \mathbf{u}_l \cdot \mathbf{u}_m. \tag{17}$$

For a whole data set X, consider all possible pairs X^2. The coassociation of all pairs is a matrix Ξ, the *coassociation matrix* (also termed *bonding relationship* in [8]). This matrix is redundant, since by definition it is symmetric. In the following, as in [40], we *serialize* the coassociation matrix, taking only the upper triangular array corresponding to its elements above the diagonal, and we obtain a coassociation vector $\mathbf{s}$ of dimension $H = N(N+1)/2$, the N-th triangular number (the number of unique pairs of entries in the matrix *including* the diagonal).

The coassociation vector is defined as

$$\mathbf{s}_h = \xi_{lm} \quad \text{for} \quad h = l(l-1)/2 + m,\ h : 1 \dots H. \tag{18}$$

when $l : 1 \dots N$ and $m : 1 \dots l$ (or, for C programmers: $h = l(l+1)/2 + m$, $h : 0 \dots H-1$ when $l : 0 \dots N-1$ and $m : 0 \dots l$). As defined, the linear index h corresponds to a row-wise scan of the lower triangular part of Ξ, including the diagonal. Note that these diagonal entries, the self-co-associations (co-associations of points with themselves), are $\xi_{ll} = 1\ \forall l$ only in non-fuzzy cases. In general, due to the triangle inequality,

$$\xi_{ll} = \sum_{j=1}^{K} u_{lj} u_{lj} = \sum_{j=1}^{K} {u_{lj}}^2 \leq \left(\sum_{j=1}^{K} u_{lj} \right)^2. \tag{19}$$

In the probabilistic case

$$\xi_{ll} \leq \left(\sum_{j=1}^{K} u_{lj} \right)^2 = 1, \tag{20}$$

where equality holds only for $u_{lj} = 1$ for some j (non-fuzzy case), whereas in the general possibilistic case

$$\xi_{ll} \leq \left(\sum_{j=1}^{K} u_{lj} \right)^2 \in \left(0, K^2\right). \tag{21}$$

However, in the AGPCM case, again

$$\xi_{ll} \leq \left(\sum_{j=1}^{K} u_{lj} \right)^2 \leq 1. \tag{22}$$

We may note that the difference between the probabilistic case and AGPCM is in the lower bound, so that for AGPCM

$$\xi_{ll} \geq 0 \tag{23}$$

but in the probabilistic case

$$\xi_{ll} \geq \frac{1}{K^2}. \tag{24}$$

Coassociations $\xi_{lm} = \mathbf{x}_l \sim \mathbf{x}_m$ (between different points) can be shown to obey similar upper bounds, while the lower bound is 0 in all cases.

3.5 Comparing Two Partitions

To compare two partitions Π^i and Π^k we compute their respective coassociation vectors $\mathbf{s}^i$ and $\mathbf{s}^k$. Note that the dimension of these vectors is H (Eq. 18), so *it only depends on the number of points, not the number of clusters.* In other words, the proposed methodology can be applied without any problem to different-size partitions.

$$\mathcal{N} = \begin{bmatrix} \mathcal{N}_{00} & \mathcal{N}_{01} \\ \mathcal{N}_{10} & \mathcal{N}_{11} \end{bmatrix} \tag{25}$$

defined by

$$\begin{aligned} \mathcal{N}_{00} &= \text{number of items s.t. } x_l \nsim_i x_m \text{ and } x_l \nsim_k x_m \\ \mathcal{N}_{01} &= \text{number of items s.t. } x_l \nsim_i x_m \text{ and } x_l \sim_k x_m \\ \mathcal{N}_{10} &= \text{number of items s.t. } x_l \sim_i x_m \text{ and } x_l \nsim_k x_m \\ \mathcal{N}_{11} &= \text{number of items s.t. } x_l \sim_i x_m \text{ and } x_l \sim_k x_m \end{aligned} \tag{26}$$

or equivalently

$$\begin{aligned} \mathcal{N}_{00} &= \| (1 - \mathbf{s}^i) \wedge (1 - \mathbf{s}^k) \|_1 \\ \mathcal{N}_{01} &= \| (1 - \mathbf{s}^i) \wedge \mathbf{s}^k \|_1 \\ \mathcal{N}_{10} &= \| \mathbf{s}^i \wedge (1 - \mathbf{s}^k) \|_1 \\ \mathcal{N}_{11} &= \| \mathbf{s}^i \wedge \mathbf{s}^k \|_1 \end{aligned} \tag{27}$$

where $\| \cdot \|_1$ is the 1-norm. Many pairwise partition similarity indexes can be practically computed starting from the contingency matrix; reference [1] provides a table.

We will also refer to the normalized contingency matrix

$$\mathcal{F} = \frac{1}{\| \mathcal{N} \|_1} \mathcal{N}. \tag{28}$$

where in the crisp case $\| \mathcal{N} \|_1 = N$. Following [11], we will use an index chosen in $\{00, 01, 10, 11\}$ to refer to generic events, where 10 and 01 refer to disagreements, 11 to a positive agreement (coassociation in both partitions) and 00 to a negative agreement (non-coassociation in both partitions).

It is simple to verify that in the general case we can compute the entries of $\mathcal{N}$ as follows:

$$\begin{array}{l} \mathcal{N}_{11} = \sum_{h=1}^{H} s_h^i \wedge s_h^k = \mathbf{s}^i \cdot \mathbf{s}^k \\ \mathcal{N}_{01} = \sum_{h=1}^{H} (\mathbf{1} - s_h^i) \wedge s_h^k = |\mathbf{s}^k|_1 - \mathbf{s}^i \cdot \mathbf{s}^k \\ \mathcal{N}_{10} = \sum_{h=1}^{H} s_h^i \wedge (\mathbf{1} - s_h^k) = |\mathbf{s}^i|_1 - \mathbf{s}^i \cdot \mathbf{s}^k \\ \mathcal{N}_{00} = \sum_{h=1}^{H} (\mathbf{1} - s_h^i) \wedge (\mathbf{1} - s_h^k) = H - |\mathbf{s}^i|_1 - |\mathbf{s}^k|_1 + \mathbf{s}^i \cdot \mathbf{s}^k, \end{array} \tag{29}$$

where $\mathbf{1}$ is an H-vector of all 1, $\cdot$ is the usual dot product and $|\mathbf{v}|_1$ is the 1-norm of vector $\mathbf{v}$. This reduces to actual counts for proper partitions; the same direct interpretation is obviously not available for fuzzy partitions, but the above definitions still hold and can be used to derive the generalized indexes.

For unsupervised learning, similarity indexes combine the off-diagonal terms of M only in commutative operations, such as products or sums, because partitions should be analysed in a symmetric fashion, since no one of them plays the privileged role of a reference. Based on this observation, to make notations more compact, we can additionally define shorthand symbols:

$$\begin{array}{rl} \pi = & \mathbf{s}^i \cdot \mathbf{s}^k \\ \sigma^i = & |\mathbf{s}^i|_1 \\ \sigma^k = & |\mathbf{s}^k|_1 \\ \sigma = & \sigma^i + \sigma^k, \end{array} \tag{30}$$

so that

$$M = \begin{bmatrix} \pi & \sigma^i - \pi \\ \sigma^k - \pi & H - \sigma + \pi \end{bmatrix}. \tag{31}$$

4 Partition Similarity Indexes

As already noted, indexes of partition similarity based on co-association, and in particular on the contingency matrix M, can be computed by several approaches. Some of them are reviewed in [31] and some are experimentally compared in [27]. Here we use loosely the term *partition* to refer to crisp partitions, fuzzy partitions, and possibilistic clusters.

4.1 The Rand and Jaccard Indexes

The Rand index [36] is defined as

$$RI = \frac{\mathcal{N}_{00} + \mathcal{N}_{11}}{\mathcal{N}_{00} + \mathcal{N}_{11} + \mathcal{N}_{10} + \mathcal{N}_{01}} \tag{32}$$

The Rand index is known to have a higher sensitivity (lower false negative rate) than specificity (higher false positive rate). This is because the index does not incorporate a-priori assumptions on a given null hypothesis, therefore is not able to distinguish false negatives from true negatives. As a result, while the index is expected to output the value 1 for identical partitions, it will not necessarily output the value 0 for non-identical partitions. To cope with this known issue, a modified version of the Rand index was proposed by Hubert and Arabie [21] incorporating a "correction for chance" which provides the ability to compare partition diversity with the null model, a hypergeometric data assumption. The adjusted Rand index is another popular choice for comparing partitions.

The Jaccard index [22] is another well-known partition similarity measure. It is defined as the ratio of the size of the intersection of two sets A and B to the size of their union:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{33}$$

and the Jaccard distance is $D_J(A,B) = 1 - J(A,B)$.

When comparing two different partitions of the same set X, this index is usually computed from $\mathcal{N}$ as follows:

$$J(\Pi^i, \Pi^k) = \frac{\mathcal{N}_{11}}{\mathcal{N}_{11} + \mathcal{N}_{10} + \mathcal{N}_{01}}. \tag{34}$$

4.2 The Fuzzy Jaccard Index

Starting from the definition of the Jaccard index, the fuzzy generalization of 34 is straightforward:

$$J_f(\Pi^i, \Pi^k) = \frac{\pi}{\sigma - \pi}, \tag{35}$$

where Π^i and Π^k are fuzzy. We call this the *fuzzy Jaccard index* [40].

The choice of the Jaccard index over other possible measures is suggested by the conclusions drawn in [43] after analysing a set of 39 different measures. The Jaccard distance $1 - J$ is a metric; the value 0 is attained only for disjoint sets; and the value 1 if and only if the two compared sets are equal.

For the fuzzy Jaccard index, the bidirectional implication in the latter property holds only for non-fuzzy sets. Therefore self-co-association gives an indication about the degree of fuzziness of a clustering. We can also define the *normalized fuzzy Jaccard index* as:

$$J_{nf}(\Pi^i, \Pi^k) = \frac{J_f(\Pi^i, \Pi^k)}{\sqrt{J_f(\Pi^i, \Pi^i) J_f(\Pi^k, \Pi^k)}}, \tag{36}$$

which is 1 when comparing a partition with itself even in the fuzzy case. Therefore, an analysis based on both J_f and J_{nf} can evaluate partition similarity and partition confidence at the same time. Note that this very natural normalization is also applied in [16], although with a different aim (make the range of values comparable between measurements).

4.3 The Fuzzy Rand Index

Campello [10] and Brouwer [8] proposed fuzzy generalizations of the Rand [36], Adjusted Rand [21] and Jaccard [22] indexes. Brouwer's proposal is based on normalized dot products between bonding relationships, i.e., cosine similarities between fuzzy membership vectors.

Proceeding in a similar way to what we did with the Jaccard index, we can define a *fuzzy Rand index*:

$$R_f() = (\Pi^i, \Pi^k) = 1 + \frac{2\pi - \sigma}{H} \tag{37}$$

and a *normalized fuzzy Rand index*:

$$R_{nf}(\Pi^i, \Pi^k) = \frac{R_f(\Pi^i, \Pi^k)}{\sqrt{R_f(\Pi^i, \Pi^i) R_f(\Pi^k, \Pi^k)}}. \tag{38}$$

4.4 The Probabilistic Rand Index

We noted earlier that the Rand index suffers from a low specificity, and that the adjusted Rand index was designed to compensate this issue. In [11] another avenue was chosen to tackle the specificity problem, by including external information in the form of weights that change the relative importance of terms in the Rand index.

The rationale for this modification is that the terms of the contingency matrix should be given different levels of relevance, since they refer to cases providing different levels of information. In particular, there have been notable discussions among the practitioners [4,9,37] about whether the number of negative matches should be taken into account at all in similarity evaluation. The Jaccard index does not take this term into account. However the Rand index RI does. A weighted version of the Rand index was therefore defined by taking into account directly the a-priori probability of the four events of interest (prior to observing the data) $c \in \{00, 01, 10, 11\}$, namely, given a pair of arbitrary data items $(\mathbf{x}_l, \mathbf{x}_m)$, the probability that they are:

– in different clusters both in Π^A and in Π^B (event $h = 00$):

$$p_{00} = \Pr(\mathbf{x}_l \nsim_A \mathbf{x}_m \text{ and } \mathbf{x}_l \nsim_B \mathbf{x}_m);$$

– in the same cluster in Π^B but not in Π^A ($h = 01$):

$$p_{01} = \Pr(\mathbf{x}_l \nsim_A \mathbf{x}_m \text{ and } \mathbf{x}_l \sim_B \mathbf{x}_m);$$

– in the same cluster in Π^A but not in Π^B ($h = 10$):

$$p_{10} = \Pr\left(\mathbf{x}_l \sim_A \mathbf{x}_m \text{ and } \mathbf{x}_l \nsim_B \mathbf{x}_m\right);$$

– in the same cluster both in Π^A and in Π^B ($h = 11$):

$$p_{11} = \Pr\left(\mathbf{x}_l \sim_A \mathbf{x}_m \text{ and } \mathbf{x}_l \sim_B \mathbf{x}_m\right).$$

Note that this definition is empirically approximated by the quantities defined in Eq. 26.

These values were computed in a maximum uncertainty (maximum entropy) hypothesis, where all clusters are equiprobable, no spatial structure is known, i.e., the probability of assigning a point to a cluster does not depend on its location, and points are uniformly sampled:

$$\begin{aligned} p_{00} &= \frac{K^A-1}{K^B}\frac{K^A-1}{K^B}; \\ p_{01} &= \frac{K^A-1}{K^A}\frac{1}{K^B}; \\ p_{10} &= \frac{1}{K^A}\frac{K^B-1}{K^B}; \\ p_{11} &= \frac{1}{K^A}\frac{1}{K^B}. \end{aligned} \tag{39}$$

Given the probability p_h of event h, the authors define a corresponding weight:

$$w_h = -\log p_h. \tag{40}$$

The *probabilistic Rand index* is defined as:

$$PRI = \frac{w_{00}\mathcal{N}_{00} + w_{11}\mathcal{N}_{11}}{w_{00}\mathcal{N}_{00} + w_{11}\mathcal{N}_{11} + w_{10}\mathcal{N}_{10} + w_{01}\mathcal{N}_{01}} \tag{41}$$

We can compute maximum likelihood a-posteriori estimates (given the data) of the probability of each of the four events of interest by approximating them with the observed relative frequencies:

$$q_h \approx f_h = \mathcal{F}_h. \tag{42}$$

By dividing numerator and denominator by the total sum, the indexes RI and PRI can be expressed using the observed frequencies f_h:

$$RI = \frac{f_{00} + f_{11}}{f_{00} + f_{11} + f_{10} + f_{01}} = f_{00} + f_{11} \approx q_{00} + q_{11} \tag{43}$$

and

$$PRI = \frac{w_{00}f_{00} + w_{11}f_{11}}{w_{00}f_{00} + w_{11}f_{11} + w_{10}f_{10} + w_{01}f_{01}} \tag{44}$$

$$\approx \frac{w_{00}q_{00} + w_{11}q_{11}}{w_{00}q_{00} + w_{11}q_{11} + w_{10}q_{10} + w_{01}q_{01}} \tag{45}$$

This formulation makes it clear that the Rand index is the probability of agreement between two partitions, after observing the data, while the probabilistic Rand index also includes correction weights that depend on the a priori probability of agreement, *before* observing the data.

4.5 The Probabilistic Jaccard Index

At this point it could be noted that the same procedure can be applied to other contingency-matrix-based indexes, like the Jaccard index J defined in Eq. 34, which can be expressed in terms of the observed frequencies/probabilities:

$$JI = \frac{f_{11}}{1 - f_{00}} \approx \frac{q_{11}}{1 - q_{00}}. \tag{46}$$

A "probabilistic" (weighted) version analogous to PRI can be defined: The *probabilistic Jaccard index* is

$$\begin{aligned} PJI &= \frac{w_{11}\mathcal{N}_{11}}{w_{11}\mathcal{N}_{11} + w_{10}\mathcal{N}_{10} + w_{01}\mathcal{N}_{01}} \\ &\approx \frac{w_{11}q_{11}}{w_{11}q_{11} + w_{10}q_{10} + w_{01}q_{01}} \end{aligned} \tag{47}$$

with the same weight definitions as per Eq. 40.

5 Applications of Fuzzy Similarity Indexes

This section illustrates some applications of the dimensionality-independent fuzzy clustering similarity indexes discussed so far. The applications include a visual technique for stability analysis and monitoring the progress of clustering by deterministic annealing.

5.1 Visual Stability Analysis Based on Comparing Fuzzy Clusterings

Stability is the tendency of a learning system to be insensitive to changes in data or in model parameters. It is related to robustness [2] and to generalisation ability [23]. As already stated, in the context of clustering it is an important quality criterion to make up for the absence of supervised information for objective evaluation. Many applications of stability in this role have been proposed [6,28,44].

Cluster model selection it is one of the most studied issues in unsupervised pattern recognition, with a long history starting in cluster analysis [36], and then borrowing ideas from robust statistics [14,18–20,32,35].

In general (see Subsect. 3.4), fuzzy similarity indexes have the property that the level of fuzziness in the partitions is reflected in the maximum value that the index can reach. This is true also for possibilistic clusters, where an added feature is that the "best" clusterings are not only the stablest, but also those with the highest degree of self-similarity (value of the similarity index when comparing a clustering to itself). Self-similarity, therefore, acts as a measure of confidence. On the other hand, we have seen also normalized indexes for possibilistic clustering, so as to eliminate this sensitivity. In this case the analysis proceeds similarly to that of probabilistic and non-fuzzy clustering. Finally, pairing the normalized and unnormalized versions of an index makes it possible to perform both sensitivity and confidence evaluations simultaneously.

Here we discuss a visual and interactive procedure, allowing the user to perceive the effect of varying one or few parameters, in this case α and β in AGPCM. Visual analysis is effective for parameters with a smooth effect on clustering performance, so we don't suggest it, for instance, to choose between different initial conditions.

We resort to a graphic representation, a heat map which includes the complete information about the distribution of the index as a function of the parameters, and suggest some criteria to evaluate this information in a visual way. The clustering similarity matrix compares every possible pair of clusterings. The matrix is symmetric, but in the possibilistic case the diagonal may contains values lower than 1: usually this indicates that the cluster centres are not significant, i.e., that during training we found a bad local minimum. Therefore, we look for values for which the diagonal is brighter. To facilitate this search when self-similarity is particularly low, we can use J_{nf}, the normalized index. However, we monitor the maximum value of the self-similarity to keep the quality of the clustering under control.

For this experiment we have chosen a data set that has a good degree of structure, but at the same time is not clearly clustered. This results in a visible instability, for instance when starting from different initialization points. The problem is provided in the base data set package of the R language and environment (www.R-project.org) as "quakes". It consists of a subset of 1000 observations of quakes (seismic events with magnitude MB > 4.0) from a larger database of 5000 observations. These quakes occurred around Fiji, starting in 1964, and are described by three-dimensional coordinates (latitude, longitude and depth of event), plus the Richter magnitude and the number of stations that reported it, for a total of 5 variables.

Since setting a large number of cluster centroids reduces instability, we kept this number relatively small, fixing it at 7. The model parameter β was swept in 9 steps in the interval $[3.9 \times 10^{-3}, 3.2 \times 10^{-2}]$. The training was performed by 9 individual runs, each with a fixed value of β.

Each individual run consisted of one random initialization, and 30 complete optimizations, each one initialized with the output of the previous one, and α sweeping from 0.1 to 1 geometrically. Another parameter is varied across the 9 individual runs of each experiment. α, starting at 0.1 and progressing up to 1, so that we obtain 30×30 similarity matrices.

The visual output of the method is shown in Fig. 2 [41]. These are plots of the value of the fuzzy Jaccard index visualized as a heat map, the clearer the higher.

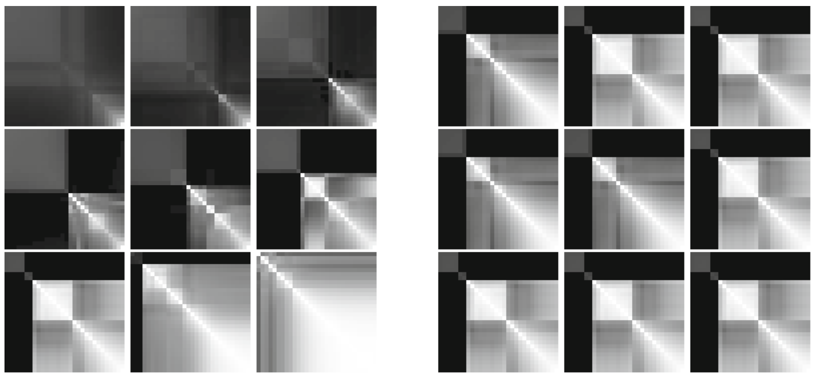


Fig. 2. Visual representation of the similarity index. Each individual heat map represents 30 experiments with different values of α ranging from 0.1 to 1. Across the 9 heat maps on the left (left to right, top to bottom): same initialization, β ranging from 0.0039 to 0.032. Across the 9 heat maps on the right: different random initializations, $\beta = 0.011$. From [41]

The first set of heat maps shows the variation as a function of β. The most stable, significant patch is attained in the seventh step ($\beta = 0.011$). The large, blurred patches in the last steps are due to the excessive value of the width parameter β. In this case, all points were attributed to a single, large cluster. On the other hand, when the width β is too small, even the diagonal has low values and only for the extreme values of α (lower right corner) data points are attributed to clusters with some confidence.

From this analysis, the best value for α is the one corresponding to the (row or column) coordinate of the center of the most stable area in the heat map. In this particular instance, the best value for α is not at the possibilistic or probabilistic extremes, but settles around an intermediate value, between 0.17 and 0.24. We select $\alpha = 0.21$.

This intermediate value is a confirmation that the Graded Possibilistic approach proposed in [30] actually provides a more flexible model, in terms of representation, than either the standard fuzzy or possibilistic methods.

If we now consider the experiments with fixed $\beta = 0.011$ and different random initializations (Fig. 2), we can see that, despite random variations in the results, the stable patch recurs in most experiments in about the same location, confirming the selected values of $\beta = 0.011$ and $\alpha = 0.21$.

5.2 Tracking Deterministic Annealing

As already noted, the Maximum Entropy clustering model described in Subsect. 2.3 is usually fit by an optimization procedure that involves gradual lowering of the model parameter. However, in contrast to the traditional Simulated Annealing [13] approach to minimization of functions of continuous variables, in this case a new annealing steps occurs only after a stable state has been reached

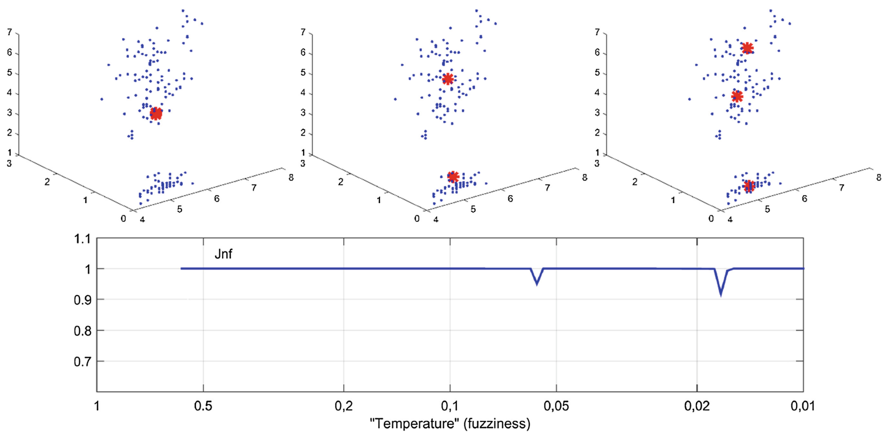


Fig. 3. Tracking deterministic annealing: Iris data.

at the previous step. This removes the main source of stochasticity, so that the method is termed Deterministic Annealing.

One peculiarity of this method is that the computational temperature parameter acts as a fuzzifier. This implies that, as the optimization progresses, less and less fuzzy clustering solutions are found, and for certain critical values of the temperature a phenomenon that parallels "phase transitions" [38] occurs. At phase transitions, cluster centroid that were overlapping because of the level of fuzziness are taken apart by the optimization. Without changing the number of centroids, this gives a hint about the number of clusters present in the data *at different resolutions.*

Fuzzy clustering comparison indices can be used to illustrate this phenomenon in high dimensionality, where the position of centroids is not easy to appreciate. But before applying the method in high dimensionality, we illustrate its operation in a lower-dimensional, well-known case, Iris data. Referring to Fig. 3, the top diagrams illustrate in three dimensions the position of centroids with respect to data in three stable states: The three centroids define one, two and three clusters depending on temperature.

The bottom diagram is a trace of the similarity of each pair of consecutive solutions, as measured by J_f and J_{nf}. In the stable states, solutions stay very similar to each other; this corresponds to flat areas where $J_{nf} = 1$. But at phase transitions, there is a sudden variation in clustering solutions. This is clearly pointed out by the notches in the graph of J_{nf}.

The high-dimensional problem chosen is the 20 Newsgroups data set. This is a collection of about 20000 Usenet posts from 20 different newsgroups. The selected version has been obtained from http://qwone.com/~jason/20Newsgroups/ already encoded by the vector space model.

To make the set more manageable only 1000 randomly selected samples from the first 5 newsgroups have been used. Due to the encoding adopted, the dimensionality has also been reduced from the original nearly 54 K to about 15 K dimensions. The number of terms in the dictionary depends logarithmically on the collection size, therefore the 1:20 reduction in data set cardinality results in less than 1:4 reduction in dimensionality and the reduced data set can still easily be categorized as high dimensional.

Figure 4 shows the result, obtained with 25 centroids. We expect about 5 clusters, so several centroids are going to overlap and we want to study the resolution (fuzziness) levels for which this overlapping is changed. Also in this case clear notches appear for quite well-defined values of the temperature parameter. By examining centroid positions in these configurations we can detect which centroids are overlapping, and how many effective clusters are there. This makes it possible to apply the multi-resolution analysis offered by the Deterministic Annealing method also in the presence of high dimensional data.

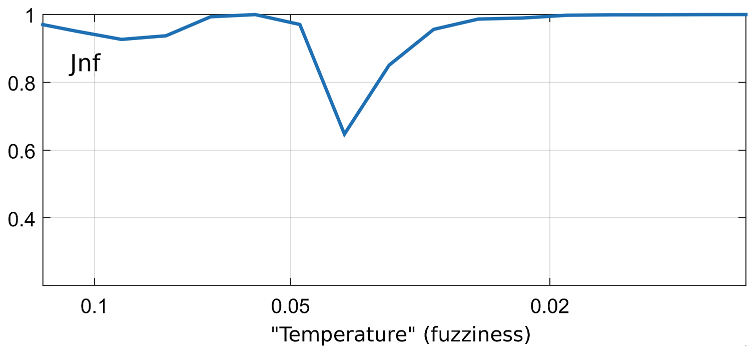


Fig. 4. Tracking deterministic annealing: 20 Newsgroups data.

6 Conclusion

In the presence of high dimensionality we face counter-intuitive situations, and visual inspection of clustering results would be beneficial. Some indices of mutual similarity between clusterings offer a way to perform this type of analysis in a dimensionality-independent way.

This chapter has presented some methods to extend these comparison indices to the cases of fuzzy and possibilistic methods. It turns out that comparing fuzzy clusterings reveals more information than in the crisp case.

In many cases we restricted the analysis to the Jaccard index, but a comparison between the possible choices from [40] could be performed.

References

1. Anderson, D.T., Bezdek, J.C., Popescu, M., Keller, J.M.: Comparing fuzzy, probabilistic, and possibilistic partitions. IEEE Trans. Fuzzy Syst. **18**(5), 906–918 (2010)
2. Anguita, D., Ridella, S., Rovetta, S.: Worst case analysis of weight inaccuracy effects in multilayer perceptrons. IEEE Trans. Neural Networks **10**(2), 415–418 (1999)
3. Barni, M., Cappellini, V., Mecocci, A.: Comments on 'A possibilistic approach to clustering'. IEEE Trans. Fuzzy Syst. **4**(3), 393–396 (1996)
4. Baroni-Urbani, C., Buser, M.W.: Similarity of binary data. Syst. Biol. **25**(3), 251–259 (1976). http://sysbio.oxfordjournals.org/content/25/3/251.abstract
5. Ben-David, S., von Luxburg, U., Pál, D.: A sober look at clustering stability. In: Lugosi, G., Simon, H.U. (eds.) COLT 2006. LNCS (LNAI), vol. 4005, pp. 5–19. Springer, Heidelberg (2006)
6. Ben-Hur, A., Elisseeff, A., Guyon, I.: A stability based method for discovering structure in clustered data. In: Altman, R.B., Dunker, A.K., Hunter, L., Lauderdale, K., Klein, T.E. (eds.) BIOCOMPUTING 2002 Proceedings of the Pacific Symposium, pp. 6–17 (2001)
7. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell (1981)
8. Brouwer, R.K.: Extending the rand, adjusted rand and jaccard indices to fuzzy partitions. J. Intell. Inf. Syst. **32**(3), 213–235 (2009)
9. Buser, M.W., Baroni-Urbani, C.: A direct nondimensional clustering method for binary data. Biometrics **38**(2), 351–360 (1982). http://www.jstor.org/stable/2530449
10. Campello, R.J.G.B.: Generalized external indexes for comparing data partitions with overlapping categories. Pattern Recogn. Lett. **31**, 966–975 (2010)
11. Carpineto, C., Romano, G.: Consensus clustering based on a new probabilistic rand index with application to subtopic retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **34**(12), 2315–2326 (2012)
12. Choi, S.S., Cha, S.H., Tappert, C.C.: A survey of binary similarity and distance measures. J. Systemics Cybern. Inf. **8**, 43–48 (2010)
13. Corana, A., Marchesi, M., Martini, C., Ridella, S.: Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. ACM Trans. Math. Softw. **13**(3), 262–280 (1987)
14. Davé, R.N., Krishnapuram, R.: Robust clustering methods: a unified view. IEEE Trans. Fuzzy Syst. **5**(2), 270–293 (1997)
15. Filippone, M., Masulli, F., Rovetta, S.: Applying the possibilistic c-means algorithm in kernel-induced spaces. IEEE Trans. Fuzzy Syst. **18**, 572–584 (2010)
16. Fowlkes, E.B., Mallows, C.L.: A method for comparing two hierarchical clusterings. J. Am. Stat. Assoc. **78**(383), 553–569 (1983). http://dx.doi.org/10.2307/2288117
17. Fred, A.L.N., Jain, A.K.: Data clustering using evidence accumulation. Int. Conf. Pattern Recog. **4**, 276–280 (2002)
18. Frigui, H., Krishnapuram, R.: A robust competitive clustering algorithm with applications in computer vision. IEEE Trans. Pattern Anal. Mach. Intell. **21**(5), 450–465 (1999)
19. Frigui, H., Krishnapuram, R.: A robust clustering algorithm based on m-estimator. In: Proceedings of the 1st International Conference on Neural, Parallel and Scientific Computations, Atlanta, USA, vol. 1, pp. 163–166, May 1995
20. Huber, P.J.: Robust Stat. Wiley, New York (1981)
21. Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**(1), 193–218 (1985)

22. Jaccard, P.: Étude comparative de la distribution florale dans une portion des alpes et des jura. Bull. Soc. Vaudoise des Sci. Nat. **37**, 547–579 (1901)
23. Kearns, M., Schapire, R.: Efficient distribution-free learning of probabilistic concepts. J. Comput. Syst. Sci. **48**(3), 464–497 (1994)
24. Klawonn, F.: Fuzzy clustering: insights and a new approach. Mathware Soft Comput. **11**(3), 125–142 (2004)
25. Krishnapuram, R., Keller, J.M.: A possibilistic approach to clustering. IEEE Trans. Fuzzy Syst. **1**(2), 98–110 (1993)
26. Krishnapuram, R., Keller, J.M.: The possibilistic C-Means algorithm: insights and recommendations. IEEE Trans. Fuzzy Syst. **4**(3), 385–393 (1996)
27. Kuncheva, L.I., Vetrov, D.P.: Evaluation of stability of k-means cluster ensembles with respect to random initialization. IEEE Trans. Pattern Anal. Mach. Intell. **28**(11), 1798–1808 (2006)
28. Lange, T., Roth, V., Braun, M.L., Buhmann, J.M.: Stability-based validation of clustering solutions. Neural Comput. **16**(6), 1299–1323 (2004)
29. Masulli, F., Rovetta, S.: Clustering High-Dimensional Data. In: Proceedings of CHDD 2012, Clustering High-Dimensional Data, Series Lecture Notes in Computer Science, LNCS 7627, 1, Springer-Verlag, Heidelberg, Germany (2015)
30. Masulli, F., Rovetta, S.: Soft transition from probabilistic to possibilistic fuzzy clustering. IEEE Trans. Fuzzy Syst. **14**(4), 516–527 (2006)
31. Meilă, M.: Comparing clusterings-an information based distance. J. Multivar. Anal. **98**(5), 873–895 (2007). http://dx.doi.org/10.1016/j.jmva.2006.11.013
32. Ménard, M., Courboulay, V., Dardignac, P.A.: Possibilistic and probabilistic fuzzy clustering: unification within the framework of the non-extensive thermostatistics. Pattern Recogn. **36**(6), 1325–1342 (2003)
33. Menger, K.: Statistical metrics. Proc. Natl. Acad. Sci. U.S.A. **28**(12), 535–537 (1942)
34. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. Society for Industrial Mathematics, Philadelphia (2009)
35. Pal, N.R., Pal, K., Bezdek, J.C.: A mixed c-Means clustering model. In: FUZZIEEE97: Proceedings of the International Conference on Fuzzy Systems, pp. 11–21. IEEE, Barcelona (1997)
36. Rand, W.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**, 846–850 (1971)
37. Real, R., Vargas, J.M.: The probabilistic basis of jaccard's index of similarity. Syst. Biol. **45**, 380–385 (1996)
38. Rose, K., Gurewitz, E., Fox, G.: A deterministic annealing approach to clustering. Pattern Recogn. Lett. **11**, 589–594 (1990)
39. Rose, K., Gurewitz, E., Fox, G.: Statistical mechanics and phase transitions in clustering. Phys. Rev. Lett. **65**, 945–948 (1990)
40. Rovetta, S., Masulli, F.: An experimental validation of some indexes of fuzzy clustering similarity. In: Di Gesù, V., Pal, S.K., Petrosino, A. (eds.) WILF 2009. LNCS, vol. 5571, pp. 132–139. Springer, Heidelberg (2009)
41. Rovetta, S., Masulli, F.: Visual stability analysis for model selection in graded possibilistic clustering. Inf. Sci. **279**, 37–51 (2014)
42. Ruspini, E.H.: A new approach to clustering. Inf. Control **15**(1), 22–32 (1969)
43. Shi, G.: Multivariate data analysis in palaeoecology and palaeobiogeographya review. Palaeogeogr. Palaeoclimatol. Palaeoecol. **105**(3–4), 199–234 (1993)
44. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters in a data set via the gap statistic. J. Roy. Stat. Soc. Ser. B Stat. Methodol. **63**(2), 411–423 (2001). http://dx.doi.org/10.1111/1467-9868.00293
45. Zadeh, L.A.: Fuzzy sets. Inf. Control **8**(3), 338–353 (1965)

Time Series Clustering from High Dimensional Data

Carlo Drago[1](✉) and Germana Scepi[2]

[1] Universitá degli Studi "Niccolo Cusano", Via Don Carlo Gnocchi 3, 166 Rome, Italy
carlo.drago@unicusano.it

[2] Department of Economics and Statistics, Universitá degli Studi di Napoli, Via Cinthia. 17, 80126 Naples, Italy
germana.scepi@unina.it

Abstract. Due to technological advances there is the possibility to collect datasets of growing size and dimension. On the other hand, standard techniques do not allow the easy management of large dimensional data and new techniques need to be considered in order to find useful results. Another relevant problem is the information loss due to the aggregation in large data sets. We need to take into account this information richness present in the data which could be hidden in the data visualization process. Our proposal - which contributes to the literature on temporal data mining - is to use some new types of time series defined as the beanplot time series in order to avoid the aggregation and to cluster original high dimensional time series effectively. In particular we consider the case of high dimensional time series and a clustering approach based on the statistical features of the beanplot time series.

Keywords: Beanplots · High dimensional data · Clustering · Self-organizing maps

1 Introduction

The growth of data results in an increased size of the modern databases. Data are collected and stored in an easier way with respect to the past. High dimensional time series are used frequently in many fields: economics and finance, bioinformatics, environmetrics and medicine [22]. In many situations it could be necessary to cope with long time series or with time series characterized by many observations not equally spaced. This is the case, for example, of high frequency data, also defined as inhomogeneous time series, particularly relevant in the financial context [5,12,14,16,25,27,28,31,33,40,41]. In all cases the need for data aggregation arises. However this data processing can lead to a loss of information so it is necessary to find different statistical techniques to handle the problem. Various authors [1–4,7,36] have proposed new approaches to the problem taking into account not only the aggregated value for the period of the single observation (by considering the day, a year and so on) but also new types of

F. Masulli et al. (Eds.): CHDD 2012, LNCS 7627, pp. 72–86, 2015.
DOI: 10.1007/978-3-662-48577-4_5

data which represent the temporal intra-period variations as intervals, boxplots or histograms. In this way the information considered is enriched by considering the entire data structure by each period and not only an aggregation. The analysis is performed by considering the symbolic data by a representation of the intra-period variation. A new approach was recently proposed in [8,10,11]. In these works we transform each temporal observation into beanplot time series so as to consider not only the minima or maxima in the period interval but also the intra-period variation. Different time series can be referred to different process characteristics, simultaneously collected [37]. In particular high dimensional time series are becoming ubiquitous in various fields. For example we can consider risk management, portfolio allocation, the study of the panel time series. At the same time it is important to remember in environmetrics the analysis of different locations and many indices in the monitoring process [22]. All these advances make use of high dimensional time series. Unfortunately these data types share some characteristics like missing values, different time series length or unequally spaced observations (such as homogeneous time series [14]). So the proposal is to consider the beanplot time series as a representation of the original time series, which allow us to take into account the intra-period variability. This transformation permits the visualization of the initial time series [11]. At this point we can consider a parameterization of the initial beanplot time series and a synthesis using the Time Series Factor Analysis (TSFA) [15] from the different time series obtained. Then from the factorial time series we extract the structural features of the factorial time series and we consider another dimensionality reduction by using the self-organizing maps [21]. The work is organized as follows: in the second section we consider the characteristics of the financial high dimensional time series; in the third we describe the beanplot time series; from the fourth to the sixth sections we illustrate our proposal: we consider beanplot time series instead of scalar time series and we also consider in this context the problem of high-dimension financial datasets. The beanplots allow us to retain the relevant information from the original time series. In particular we start from the choice of the temporal interval and from the visualization of the original data. So we obtain the transformation in beanplot time series, then we start with the interpretation of the data. In section seven we consider an approach based on clustering these types of time series based on self-organizing maps. The remaining part of the work is devoted to a simulation study (with two different synthetic datasets) and the application to real financial data.

2 Financial High Dimensional Data Characteristics

High dimensional time series are a relevant issue in present day data: [6,17,22,29,35]. In the financial context high dimensional time series can be, for example, referred to different stocks (asset prices), exchange rates or different financial items. High dimensional data are becoming more and, more importantly in economic and financial applications and nowadays represent a big challenge. At the same time financial time series present relevant characteristics. In particular they are characterized by high volatility, missing values and series of unequal length (for a review of characteristics of the financial time series [32]). Data representing each different financial item (for example an asset price) simultaneously considered in the time can be collected in different vectors. The concept of high dimensional data in our case is directly related to the number of observations in typical high frequency financial data. In these cases the observations are overwhelming and so it is necessary to consider these data by using this method in the appropriate way. In particular it is necessary to take into account the specific techniques when there are longer time series and it is necessary to handle high dimensional financial time series. In these cases classical clustering techniques become impractical [38] and it is necessary to take into account different techniques. There are various proposals in this respect [26,38] to use a new method which is useful in the case of high dimensional data. Replacing the original time series with some measures of the characteristics can be used in many cases [38]. At the same time new problems have to be considered in the case of large datasets. In financial time series, for example, aggregation by a single temporal interval can lead to a loss of information. This is the case with the High-Frequency data [14] in which the number of observations in the single day is usually overwhelming. There are various proposals and approaches to cope with this problem in this context. In particular many proposals belong to the field of the Symbolic Data Analysis literature [4]. More in general the purpose of this literature is to represent the intra-period data as an aggregate representation such as Intervals, Boxplots, Histogram time series. In this context the original time series to be analyzed are not related to scalars but are related to the complex objects or the symbolic data [1]. These different types of time series show as their principal characteristic the capability to take into account not only the inter-temporal variability (like the scalar time series) but also the intra-temporal variability. Drago and Scepi [10] show that these time series tend to retain the information of the original scalar time series (the trend, the seasonality and the cycle for example) and allow to compare quickly the variability of the series over time and more importantly the structural changes. In this case by considering the motivating problems (financial high dimensional time series) when the number of time series is high we need appropriate approaches to take into account a higher number of columns. The aim of this work is to innovate the temporal data mining approaches in these types of time series. In particular, different approaches have been proposed and studied in recent years considering some types of representations based on histograms [2]. We propose an innovative approach of clustering high dimensional data based on the Beanplot Time Series to avoid information loss and to take into account the intra-period variability in the clustering process. In this work we will present the methods and the relevant code in R [30] to replicate the methods for other real cases.

3 Beanplot Time Series

The aim of clustering is to obtain different homogeneous groups from an unlabeled data set of time series i where the dissimilarity within groups needs to be maximized [24]. We start from a data set of time series i.

$$P_i = \{P_{i_t} : t \in T\} \tag{1}$$

Where i is a different index representing the time series, and t is the time. To avoid the problem of data aggregation in very long time series, various authors propose different representations like histograms, intervals or boxplots. We propose a different representation taking into account the intra-period variation [8,10]. A Beanplot Time Series (based on Beanplot [19]) can be defined as an ordinated sequence of beanplot data over time t.

$$\hat{f}_{h,t} = \frac{1}{nh}\sum_{n}^{i=1} K(\frac{x - x_i}{h}) \tag{2}$$

Here K is a defined kernel and h can be defined as a smoothing parameter or a bandwidth. K can be a Gaussian function with the mean 0 and variance 1. An important property of the kernel density estimation is that the area under the curve is 1. Thus we obtain the Fig. 1 for the beanplot time series related to the US Market (period 1996–2010).

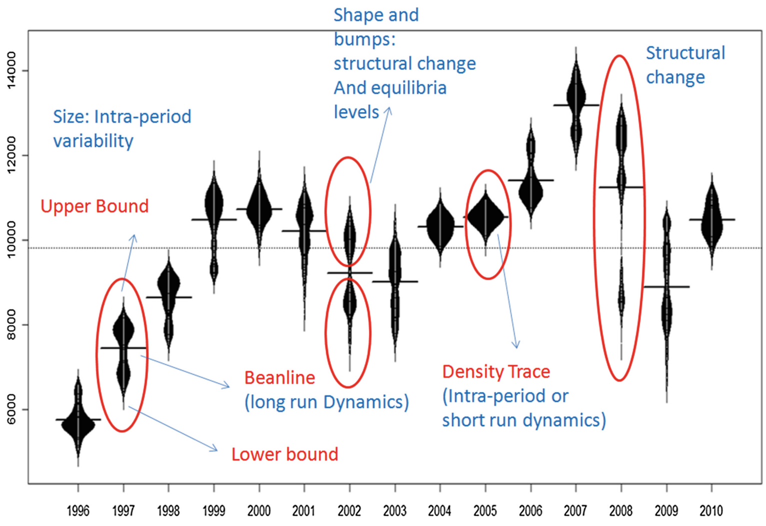


Fig. 1. Beanplot time series

Figure 1 shows that the single beanplot typically represents the variation intra-period, where the beanplot dynamics shows the variation inter-period during the time. Upper and Lower bound indicates the minima and the maxima where the density trace shows structural change over time. The bumps show equilibria levels over times [11], where the beanlines show the aggregated values of the series (the mean, or the meadians by period). The kernel is typically a non-negative and real-valued function $K(u)$ which satisfy [23]:

$$\int K(u)du = 1, \int uK(u)du = 0, \int u^2K(u) = k_2 < \infty \tag{3}$$

The lower and upper limits of integration being $-\infty$ and $+\infty$. In general various Kernel function can be normally used, like the uniform, triangle, Epanechnikov, quartic (biweight), tricube (triweight), Gaussian and cosine. In particular the h parameter controls the variance. We can establish the parameter h by choosing the Sheather-Jones method [34].

$$K(\frac{x-x_i}{h}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x-x_i^2}{2h^2}} \tag{4}$$

The beanplot beanline can be specified as the mean or median. Relevant characteristics of the beanplots (in particular as representing the variability) are the beanplot upper and the lower bound:

$$[X]_t = [X_{t,L}, X_{t,U}] \quad \text{where } -\infty < X_{t,L} \leq X_{t,U} < \infty \tag{5}$$

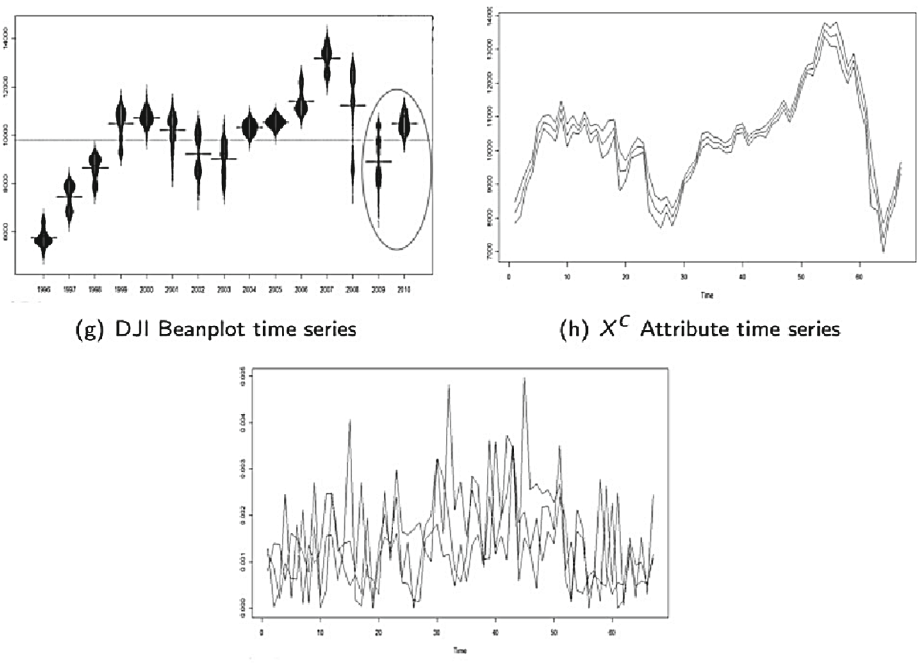
Another way to represent the information related to the location and the size of the Beanplots are the Beanplots Center and Radius:

$$[X]_t = \langle X_{t,C}, X_{t,R} \rangle \text{ where } X_{t,C} = (X_{t,L} + X_{t,U})/2 \text{ and } X_{t,R} = (X_{t,U} - X_{t,L})/2 \tag{6}$$

In this sense we can have information both for the long run dynamics (represented by the beanlines over time) and the intra-day variability represented by the upper and lower bounds and centers and radius. The density trace shows the entire data structure at time t. An important choice is related to the interval period I to define each beanplot. In this sense we need to consider a spectral analysis before the analysis to avoid hiding the cycles. Another possibility is to define an a priori choice for the specific window depending on the scope of the analysis.

4 Parameterizing Beanplot Time Series Data

The beanplot time series need to be parameterized in order to define and compare the different beanplots over time. From the original data we obtain the trajectories related to the beanplots (also defined as attribute time series). For each parameter we obtain an attribute time series. Then after a process related to the TSFA [15] we obtain the factorial time series z_t. So it is relevant to choose adequately the bandwidth h for the beanplot time series and the number of parameters or features considered n. In particular we need to choose a unique h parameter for the entire beanplot series. It is important to note that in the use of the beanplot time series in visualization aims [11] or in an exploratory data analysis context there is no need to choose a unique h parameter by the Sheather-Jones method [34] or choose another criteria. In fact in the parameterization of the different attribute time series for each beanplot there is the need to take into account the differences between the beanplots [10]. However using adequately the attribute time series it is possible to detect outliers in the entire beanplot time series. We need to define adequately the process described by the beanplots, so we need to choose accordingly the parameters X^c and Y^c, its number n and the bandwidth h in order to represent correctly the beanplots and the underlying data. In particular the number of parameters n represents both the data structure of the interval temporal and the beanplot evolutive dynamics over time by means of its attribute time series. As an output of the process we note that the X^c and the Y^c can be differently intepreted by considering comparatively different beanplots: the long run data structure for the X^c and the variability for the Y^c. From the beanplot time series we obtain attributes time series for X^c and the Y^c. In particular we show an example in Fig. 2 for the beanplot attribute time series of the Dow Jones Index 1996–2010, in which we have chosen $n = 3$ for X^c and Y^c).



(g) DJI Beanplot time series

(h) X^C Attribute time series

(i) Y^C Attribute time series

Fig. 2. Attribute time series

The crucial point in the parametrization process is the choice of the bandwidth h and the choice of the number of the n features considered in relation to the data structure. It is important to validate the choices in the parameterization (the n choice of the features and the chosen bandwidth h) and its adequacy to represent initial data. In this way the validation occurs in repeating the process a number of times to find an optimal representation of the beanplot time series and its parameters. In general it is necessary to take into account a lower bandwidth h in the series with a higher number of features n if there is a higher level of observations: thus, we can capture a higher number of features. In particular, we consider that the higher the complexity of the original time series, the higher the complexity requested and the number of features to be taken into account. Therefore, it is necessary for most of the cases to reproduce the structure of the beanplot by considering at least three descriptors for the X^c and three descriptors for the Y^c.

5 Time Series Factor Analysis on Beanplot Time Series

From the attribute time series identified in the parameterization process we need to synthetyze the attribute time series. These factorial time series substitute the original one and can be used in the clustering process. In particular we use the Time Series Factor Analysis (TSFA) depicted in [15] to estimate to the attribute time series for the X^c and for the Y^c. We start from the n observed processes $z_{i,t}$ with $i = 1..n$ and $t = 1..T$, where we search from the k unobserved processes (the factors) $\xi_{i,t}$ with $t = 1..T$ and $i = 1..k$ to obtain the measurement model:

$$z_t = \alpha + B\xi_t + \epsilon_t \tag{7}$$

Where α is a vector of intercepts, B is an $n \times k$ matrix of factor loadings and ϵ is a n vector of random errors. Each factor score represents a measurement model of a latent variable that is the underlying phenomena of interest. At the end of the procedure we obtain a set of factors for each attribute time series. Following [15] for measuring the factor score predictor we use the Bartlett predictor.

$$\xi_t = (B^t \Psi_t^{-1} B)^{-1} B^t \Psi^{-1} B^t (z_t - \alpha_t) \tag{8}$$

Where: ψ_t is the covariance of (ϵ_t). The loadings can be estimated by Factor Analysis estimators (Maximum Likelihood in particular) using the sample covariance of the error [15]. We compute one factor time series, we define from now on as z_t for the X^c and for the Y^c, using the attribute time series. The two factorial time series X^c and Y^c represent the general dynamics of the beanplot (in particular the location and the size), and the second one represents the response of the shock or the short run dynamics (the shape). The final aim of the clustering process is to recognize groups of time series with a synchronous dynamics (related to the location of the beanplot or the X^c) and a similar response to the shocks (related to the Y^c).

6 From Time Series Factor Analysis to the Feature Clustering Approach

The aim is to cluster the different beanplot time series by using the different factorial time series. In Finance, clustering techniques can be applied in a range of problems like Asset Allocation and Statistical Arbitrage. Clustering, in particular, can be defined as the process of obtaining different groups of items (or time series) highly similar between themselves but different by groups [24]. Liao (2005) [24] reviews the different approaches in clustering time series. Given a set of time series it is necessary to have a clustering algorithm or a procedure to classify a set of unlabelled time series. The specific choice of the algorithm can depend on different factors and different data typologies, for example the different purposes and application [24]. At the same time there can be different distinctions which it is possible to make by considering different time series. Liao [24] makes a list of relevant distinctions in time series clustering: discrete-valued data or real valued, uniformly and non-uniformly sampled data, univariate and multivariate and equal or unequal length, but Kavitha and Punithavilli consider clustering techniques of data streams [20]. At the same time non uniformly sample data need to be transformed by obtaining uniformed data [38]. In these cases there can be sometimes the need for an aggregation process, which leads to loss of information, for example in the case of clustering of high frequency data [14]. Another approach is to consider a representation which does not lose the information on intra day variability. In this context it is possible to consider all the works in Symbolic Data Analyis (SDA) [3,4]. So in this sense we cluster the different representations of the series such as intervals, boxplots, or histograms of the series with the aim of taking into account the intra-period variability. By considering the beanplot representation [9,10] and its parameterization in attribute time series [8,11] we start from the factorial time series which represents each time series then we cluster directly the factorial time series. So we can have many columns related to a single process or phenomenon and we need to classify groups of homogeneous factorial time series. In these cases the classical techniques are difficult to be implemented and there is the need to consider some other new techniques to take into account the different data characteristics [38]. We can start from the set of factorial time series we have obtained, which represents the initial time series. So we obtain the factorial time series for the X^c and the Y^c then we compute the dissimilarity matrix. In particular, correlational distances can be used as distances. We try to specifically recognize the correlation between the dynamics of the different synthesizing factors over time. So we have for two generic factorial time series a and b:

$$d(a,b) = 1 - (corr(\xi_a, \xi_b)) \tag{9}$$

Considering the absolute values:

$$d(a,b) = 1 - |(corr(\xi_a, \xi_b))| \tag{10}$$

and finally:

$$d(a,b) = \sqrt[2]{1 - (corr(\xi_a, \xi_b))^2} \tag{11}$$

Where it is possible to recognize the correlation between the dynamics of the different synthesizing factors over time [8]. It is possible to obtain the dissimilarity matrices related and we use different clustering methods to compare the different results. In particular it is possible to use the hierarchical clustering and the non hierarchical clustering by using different methods. At this point we can apply different methods to observe the robustness of the results. However for long factorial time series this approach is not simple [38]. So we can consider an approach for considering very long Beanplot time series which allow us to consider the unequal length of the factorial time series. In practice from each factorial time series we extract the characteristic features [26,38]. Following [38] the features we consider represent the characteristics of the time series. These are chosen by considering extensively all the characteristics of the factorial time series, such as trend and seasonality, periodicity, serial correlation, non linear autoregressive structure, skewness, kurtosis and so on [18,38]. The features represent the stylized representation of the original factorial time series and they are scaled from 0 to 1 to show if a feature is particularly strong compared to the considered factorial time series [38]. In this way we can explore structural similarities between the series, considering not the actual values but the structural characteristics of the factorial time series. In the clustering algorithm they represent the finite set of inputs for the Self Organizing Maps (SOM) [13]. We consider the features from the factorial time series. The computations can be obtained using the package Kohonen in R [39].

7 Using the Self Organizing Maps

We propose an alternative clustering approach to cluster. In this way most of the clustering is done interactively not in an automated fashion. We can use the Self Organizing Maps to reduce the data dimensionality. In this sense the input of the data matrix are the factorial time series characteristics and then the output are the labels for each beanplot time series. Self Organizing Maps can be considered a typology of neural networks with neurons organized as a low dimensional structure and trained in an iterative unsupervised procedure [24]. A SOM is characterised by a nonlinear projection of a high dimensional data mostly on a two dimensional low grid [38]. The learning algorithm of the SOM is initialized by using the same scheme used in all the neural networks. So the training process is carried out by presenting the input patterns by assigning small random values to the neurons weight vectors w in the network [24]. In this way each input pattern s is presented to all grid points where the closest matching to the weights of each neuron l is found:

$$d_l = \|s - w_l\| = \sqrt{\sum_{l=1}^{n}(s_i - w_{il})^2} \tag{12}$$

Then the neuron output is computed and the weighting w is updated [38]. The results of the clustering approach usually show two aspects: the data classes or labels and the metric or topological relations between the initial data [38]. In this case the input is represented by the features extracted from each factorial time series representing each beanplot time series. From the training process we can obtain the output of the SOM in the form of a two dimensional map visualizing the groups of the factorial time series. The clustering of the factorial time series is the main output.

8 Simulation Study

Now we will experiment the validity of the method by considering 2 distinct computational experiments. We consider different simulated factorial time series from the original beanplot time series. In particular we simulate 125 different time series. Each time series is characterized by 96000 observations, and they are originally transformed into 2400 temporal beanplots to represent the data structure accordingly. From the beanplot time series we parameterize, considering three parameters: three for X^c and three for Y^c. It is important to note that we consider three points for X^c and three points for the Y^c which seems to cover the aspects we want to take into account. The bandwidth for the entire series is fixed at 3.14, a value obtained after the search of the best relevant value for the entire series. So in this case, following the Time Series Factor Analysis methodology we obtain the 125 factorial time series related to the factor 1 (one factorial time series for each beanplot). The factor 1 represents the long run dynamics of the beanplot time series, so we are mainly considering the long run dynamics of our time series. At this point we extract the characteristic features of factorial time series, then we consider the matrix obtained by the characteristic features as input of the SOM. The result is shown in Fig. 3.

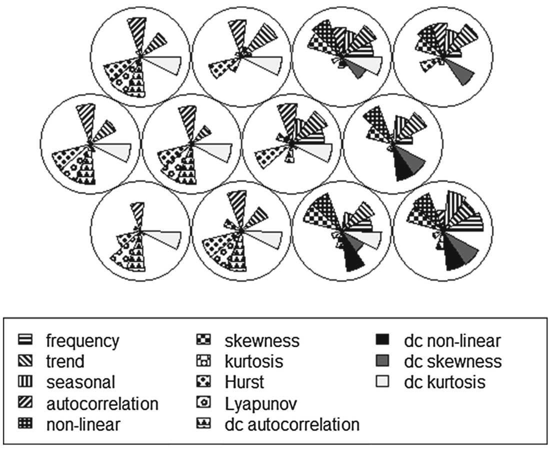


Fig. 3. Simulation 1: 128 Beanplot time series

In Fig. 3 the codebook vectors are visualized in a specific segment plot [39]. By observing the image it is possible to see the associations between the different characteristics of the factorial time series. In particular we can observe that the time series are regrouped by similar characteristics and the clusters show the relevant characteristics of the group. The plot of the codebook vectors shows a mapping of the data (in particular considering the factor 1 of the factorial time series). We found mainly three groups in the analysis. A group of beanplot time series with features in the factorial time series show higher Hurt, Lyapunov and

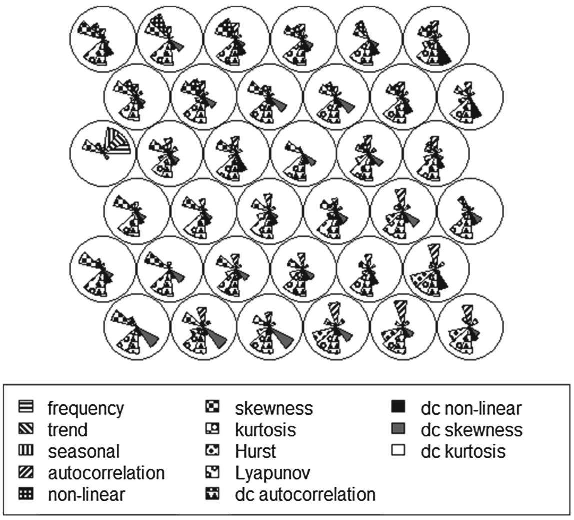


Fig. 4. Simulation 2: 500 Beanplot time series

dc autocorrelation coefficient (dc is for decomposition) and also higher seasonal characteristics. Another group is characterized by dc nonlinear and also dc skewness, whereas a third group is characterized by various general characteristics from nonlinear to frequency. So we obtain the ex-post clusters and we compare them with the expected initial groups. We repeat the same procedure by using other clustering methods to compare the results. In particular we use the correlation distance between the factorial time series and the Hierarchical Clustering (correlation distance, complete linkage method) and the classical K-Means. We compare the clusters obtained also by modifying the parameters of the SOM. It is important to train different maps to observe the sensitivity of the results to changes [39]. The results of the ex-post results confirm our ex-ante expectations. We compare the results obtained considering all the replications using the different methods (SOM on characteristic features, Hierarchical Clustering using correlation distance and classical K-Means).

In experiment 2 we are directly simulating from the results of a factorial time series whereas in experiment 1 a number of 500 different factorial time series are considered. Results for the trained network are in Fig. 4. In this case we observe a more similar structure for the beanplot time series and the features of the factorial time series representing them. In particular the similar characteristics are Lyapunov and dc autocorrelation for all the groups. Differences are related mainly to single characteristics like trend or skewness. There is a strange group characterised by frequency, trend and seasonal. That means there exists some factorial time series which behave strangely. The results are gained when we have simulated some of the factorial time series starting from some measured ones and then performing some model changes.

9 Application on Real Data

In the application we consider a set of 100 random stocks on the US market (the period 1900–2012 is considered). It is important to note that we are using daily data but that the methods can also be used for longer time series. However the method presented in the paper considers the feature of the time series so could be used also for inhomogeneous time series. In any case we maximize the number of different observations considered and so we consider series of different length. In this way we obtain the beanplot time series, then we extract the characteristics of the series to compare with the long run dynamics.

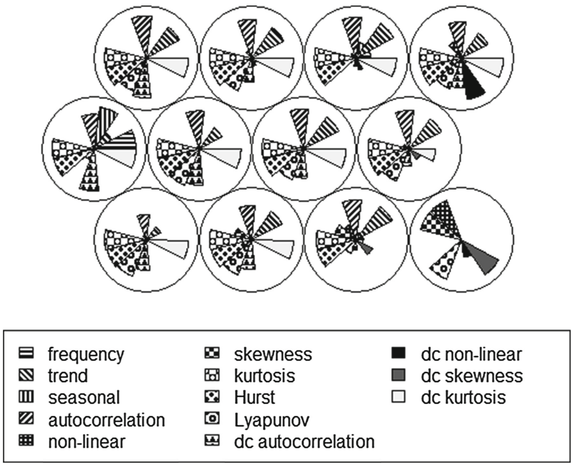


Fig. 5. Application on financial data (US Stock Market)

The results for the trained network are visualized in Fig. 5. The patterns show the different characteristics for the groups (by observing the different codebook vectors). The classification of the single factorial time series fit the initial expectations. By considering the different clusters obtained there is evidence of the different market mechanisms of shocks transmission between the different stocks. The different characeristics for each codebook vectors of the networks (see Fig. 5) associated to a group of series show the common features and behavior of the series. So the difference between the SOM units has relevant economic implications. In particular we observe that stocks also act as a signal for the entire market. It is interesting to observe the similarity between the different codebook vectors. By observing the codebook vectors we can visualize the different patterns in the data. At the same time it is interesting to focus on specific characteristics of the factorial time series (like the beanplots) to exploit them in statistical arbitrage models. So in this sense the procedure could be extremely interesting in exploring possible couples or triples of stocks with the aim of making statistical arbitrage using financial data. In particular by observing the similar stocks in the groups (stocks reacting similarly to the financial shocks) we can find important patterns to exploit in the trading models.

10 Conclusions

We have considered in this work an approach to handle high dimensional time series by means of beanplot time series. The results are of particular interest when it is remembered that we are considering very long time series, so we are using a higher quantity of information. The modern datasets are characterized by a high quantity of information so it is essential to have a technique to handle this data in an adequate way. In particular the extraction of the features for the different time series shows the relevant characteristics useful in financial applications, for example, where we show the way to transform the results into specific decisions at an operational level.

References

1. Arroyo, J., Gonzales Rivera, G., Maté, C.: Forecasting with Interval and Histogram Data: Some Financial Applications. Working Paper (2009)
2. Arroyo, J., Maté, C.: Forecasting histogram time series with K-nearest neighbours methods. Int. J. Forecast. **25**, 192–207 (2009)
3. Billard, L., Diday, E.: From the statistics of data to the statistics of knowledge: symbolic data analysis. J. Am. Stat. Assoc. **98**, 991–999 (2003)
4. Billard, L., Diday, E.: Symbolic Data Analysis: Conceptual Statistics and Data Mining. Wiley, Chichester (2006)
5. Brownlees, C.T., Gallo, G.M.: Financial econometric analysis at ultra-high frequency: Data handling concerns. Comput. Stat. Data Anal. **51**(4), 2232–2245 (2006)
6. Deistler, M., Zinner, C.: Modelling high-dimensional time series by generalized linear dynamic factor models: an introductory survey. Commun. Inf. Syst. **7**(2), 153–166 (2007)
7. Diday, E., Noirhomme, F.: Symbolic Data Analysis and the SODAS Software. Wiley-Interscience, Chichester (2008)
8. Drago, C.: The Density Valued Data Analysis in a Temporal Framework: The Data Model Approach. Ph.D Dissertation in Statistics XXIV Cycle, University of Naples "Federico II" (2011)
9. Drago, C., Lauro, C., Scepi, G.: Visualizing and Forecasting Beanplot Time Series Working Paper (2010)
10. Drago, C., Scepi, G.: Forecasting by Beanplot Time Series Electronic Proceedings of Compstat. Springer Verlag, pp. 959–967 (2010). ISBN 978-3-7908-2603-6
11. Drago, C., Scepi, G.: Visualizing and exploring high frequency financial data: beanplot time series forthcoming. In: Ingrassia, S., Rocci, R., Vichi, M. (eds.) New Perspectives in Statistical Modeling and Data Analysis, Springer Series: Studies in Classification, Data Analysis, and Knowledge Organization (2011). ISBN: 978-3-642-11362
12. Engle, R.F., Russell, J.R.: Analysis of high frequency financial data. In: Hansen, L., Ait-Sahalia, Y. (eds.) New York, vol. 6(1), pp. 47–53 (2004)
13. Fu, T.-C., et al.: Pattern discovery from stock time series using self-organizing maps. In: Workshop Notes of KDD 2001 Workshop on Temporal Data Mining, pp. 26–29 (2001)
14. Gençay, R., et al.: An Introduction to High Frequency Finance, 1st edn. Academic Press, San Diego (2001)

15. Gilbert, P.D., Meijer, E.: Time Series Factor Analysis with an Application to Measuring Money. Research Report No. 05F10. University of Groningen, SOM Research School (2005)
16. Glattfelder, J.B., Dupuis, A., Olsen, R.B.: Patterns in high-frequency FX data: Discovery of 12 empirical scaling laws. Quant. Finance **11**(4), 26 (2008)
17. Heaton, C.: Factor Analysis of High Dimensional Time Series. Ph.D. Thesis, University of New South Wales. School of Economics, Sydney, Australia (2008)
18. Hyndman, R.J.: Measuring Time Series Characteristics. R-Bloggers (2012)
19. Kampstra, P.B.: A boxplot alternative for visual comparison of distributions. J. Stat. Softw. **28**(1), 1–9 (2008). Code Snippet 1
20. Kavitha, V., Punithavalli, M.: Clustering time series data stream: a literature survey. J. Comput. Sci. **8**(1), 289–294 (2010)
21. Kohonen, T.: Self-Organizing Maps Springer Series in Information Sciences, vol. 30. Springer, Heidelberg (2001)
22. Lam, C., Yao, Q., Bathia, N.: Estimation for latent factor models for high-dimensional time series. Biometrika **98**, 35 (2010)
23. Li, Q., Racine, J.S.: Nonparametric Econometrics: Theory and Practice. Princeton University Press, Princeton (2006)
24. Liao, W.T.: Clustering of time series data a survey. Pattern Recogn. **38**, 1857–1874 (2005)
25. Myland, P.A., Zhang, L.: The Econometrics of High Frequency Data. Working Paper (2009)
26. Nanopoulos, A., Alcock, R., Manolopoulos, Y.: Feature-based classification of time-series data. Int. J. Comput. Res. **10**, 49–61 (2001). Nona Science
27. OlsenWorld: High Frequency Data (2012). http://www.olsen.ch/more/datasets/
28. Pasley, A., Austin, J.: Distribution forecasting of high frequency time series. Decis. Support Syst. **37**(4), 501–513 (2004)
29. Rao, R.B., Rickard, S., Coetzee, F.: Time series forecasting from high-dimensional data with multiple adaptive layers. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD 1998), pp. 319–323 (1998)
30. R Development Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2010). ISBN 3-900051-07-0
31. Russell, J.R.: Econometric Modeling of Multivariate Irregularly-Spaced High-Frequency Data. Working Paper (1999)
32. Sewell, M.: Characterization of financial time series, Research Note RN/11/01. University College London, London (2011)
33. Sewell, M.V., Yan, W.: Ultra high frequency financial data. In: Proceedings of the 2008 GECCO Conference Companion on Genetic and Evolutionary Computation GECCO 2008, pp. 18–47 (2008)
34. Sheather, S.J., Jones, M.C.: A reliable data-based bandwidth selection method for kernel density estimation. JRSS-B **53**, 683–690 (1991)
35. Taylor, G.: Compact Modeling of High-Dimensional Time Series (2007)
36. Verde, R., Irpino, A.: Comparing Histogram Data Using a Mahalanobis Wasserstein Distance P. Brito, ed. Analysis **2008**, 77–89 (2008)
37. Verleysen, M., François, D.: The curse of dimensionality in data mining and time series prediction. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 758–770. Springer, Heidelberg (2005)
38. Wang, X., Smith, K.A., Hynman, R.J.: Characteristic based clustering for time series data. Data Mining Knowl. Discov. **13**(3), 335–364 (2006)

39. Wehrens, R., Buydens, L.M.C.: Self- and Super-organising Maps in R: the kohonen package. J. Stat. Softw. 21(5) (2007). http://www.jstatsoft.org/v21/i05
40. Yan, B., Zivot, G.: Analysis of High-Frequency Financial Data with S-PLUS. Working Paper (2003)
41. Zivot, E.: Analysis of high frequency financial data: models, methods and software. Descriptive Analysis of High Frequency Financial Data with S-PLUS. Presented at the 11th Brazilian Time Series and Econometrics Meeting (ESTE), July 31–August 3 (2005)

Data Dimensionality Estimation: Achievements and Challanges

Francesco Camastra(✉)

Department of Science and Technology, University of Naples Parthenope,
Centro Direzionale Isola C4, 80143 Naples, Italy
camastra@ieee.org

Abstract. Dimensionality Reduction methods are effective preprocessing techniques that clustering algorithms can use for coping with high dimensionality. Dimensionality Reduction methods have the aim of projecting the original data set of dimensionality d, minimizing information loss, onto a lower M-dimensional submanifold. Since the value of M is unknown, techniques that allow knowing in advance the value of M, called intrinsic dimension (ID), are quite useful. The aim of the paper is to make the state-of-art of the methods of intrinsic dimensionality estimation, underlining the achievements and the challanges.

1 Introduction

Dimensionality Reduction methods are effective preprocessing techniques that clustering algorithms can use for coping with high dimensionality. Dimensionality Reduction methods have the aim of projecting the original data set $\Omega \subset \mathbb{R}^d$, minimizing information loss, onto a lower M-dimensional submanifold of $\mathbb{R}^d$. Since the value of M is unknown, techniques that allow knowing in advance the value of M, are quite useful. Following Fukunaga, a data set $\Omega \subset \mathbb{R}^d$ is said to have *intrinsic dimension* (ID) [16] equal to M if its elements lie entirely within a M-dimensional submanifold of $\mathbb{R}^d$, where $M < d$. It is important to observe that ID depends on the scale of data. In order to show this, it considers a two-dimensional data set, e.g., a K-Möbius strip [20], adding to a data set a three-dimensional gaussian noise. The data set, obtained in this way, has ID equal to 2 at a coarse scale, since the two-dimensional set is dominant. But if we change scale and observe the data set at fine scale, the noise becomes dominant and the ID of data set is three. ID estimation of a data set is a classical problem of pattern recognition and machine learning. The first algorithm of data dimensionality estimation, by Bennett, dates back to 1969 [3]. ID estimation is relevant in machine learning not only for dimensionality reduction methods but also for other several reasons. Firstly, using more dimensions than the necessary leads to several problems, such as an increase of the space required to store data, a decrease in the algorithm speed, since it generally depends on the data dimensionality. Besides, building reliable classifiers becomes harder and harder when the dimensionality grows (*curse of dimensionality* [2]). To this purpose,

F. Masulli et al. (Eds.): CHDD 2012, LNCS 7627, pp. 87–101, 2015.
DOI: 10.1007/978-3-662-48577-4_6

we recall that the capacity (*VC-dimension*) [57] of the linear classifiers, that determines their generalization capability, may depend on ID. Finally, ID is relevant for some prototype-based clustering algorithms. For example, ID affects the *magnification factor* [59] of a trained Neural Gas, that expresses the relation between the data density and the density of the neural gas weight vectors[1].

The aim of the paper is to make the state-of-art of the methods of the intrinsic dimensionality estimation, underlining the advances and the open problems. Extending the taxonomy proposed by Jain and Dubes [25], we group the algorithms for estimating ID in three disjoint categories, i.e., *local*, *global*, *mixed.* In the local category, there are the algorithms that provide an ID estimation using the information contained in sample neighborhoods. To the global category belong the algorithms that make use of the whole data set providing a unique and global ID estimate for the data set. Finally, in the mixed category, there are the algorithms that can produce both a global ID estimate of the whole data set and local ID estimate of particular subsets of the data set. In the paper the most relevant algorithms for each category, underlining their weak points, will be presented. In particular, it will be discussed the robustness of each method with respect to the high dimensionality. The paper is organized as follows: Sects. 2, 3, 4 describe global. local and mixed methods, respectively; the benchmarking of ID estimation method is discussed in Sects. 5 and 6 open problems are analyzed and some conclusion are drawn.

2 Global Methods

In the global category, the algorithms unfold the data set in the d-dimensional manifold. Unlike local methods that use only the information contained in the neighborhood of each data sample, global algorithms make use of the whole data set. These methods make implicitly the assumption that the data lie on a unique manifold of a fixed dimensionality. Global methods can be grouped in four families: *projection techniques*, *fractal-based algorithms*, *multidimensional scaling methods* and *other techniques*, where in the last category are collected all the methods that cannot be assigned to the first three categories.

2.1 Projection Techniques

Projection techniques search for the best subspace to project the data by minimizing the projection error. *Principal Component Analysis* (*PCA*) [26,30] is the simplest and the most widely used projection method. PCA is a linear projection method since projects the data along the directions of maximal variance. PCA algorithm for ID estimation has the following steps:

1. Compute the N eigenvalues of the covariance matrix. Order them in decreasing way, such that $\lambda_1 \geq \lambda_2, \cdots \geq \lambda_N$.

[1] If we denote with P the relation between the data density P and the density ρ of the weight vectors, then $\rho \propto P^{\alpha}$ where $\alpha = \frac{ID}{ID+2}$.

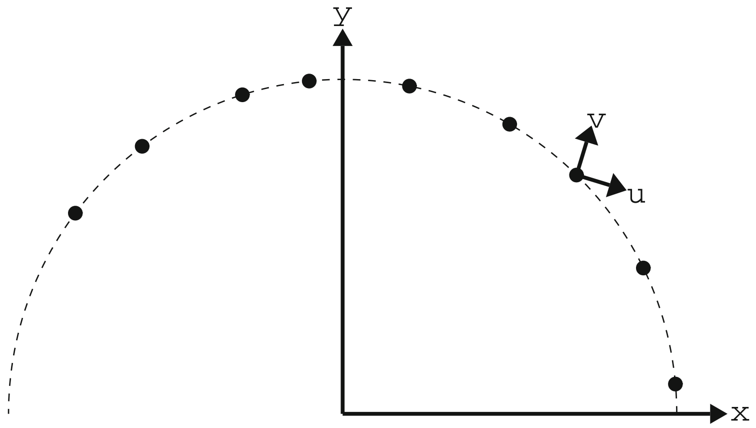


Fig. 1. *Ω* Data set. The data set is formed by points lying on the upper semicirconference of equation $x^2 + y^2 = 1$. The ID of *Ω* is *1*. Neverthless PCA yields *two* non-null eigenvalues. The principal components are indicated by *u* and *v*.

2. Normalize the eigenvalues dividing each eigenvalue by the largest one λ_1.
3. Choose a threshold value θ and compute the integer K such that $\lambda_K \geq \theta$ and $\lambda_{K+1} < \theta$.
4. return (ID=K).

It is easy to show that the loss of the information due to the discarding the lowest (N-K) eigenvectors is equal to the sum of the lowest (N-K) eigenvalues [4]. PCA is a poor estimator, since it tends to overestimate the ID. Consider a data set formed by datapoints lying on a circumference, (Fig. 1) PCA yields an ID estimate equal to 2 instead of the correct value of 1. Therefore we can assess that, since PCA overestimates ID, PCA provides can be an upper bound of the actual ID value of a dataset. Nonlinear projection methods have been designed in order to overcome the PCA limitations. In order to cope with these problems, some algorithms have been proposed to get Nonlinear PCAs. A widely used approach to get a Nonlinear PCA is the autoassociative approach [28]. Nonlinear PCA is performed by means of a five-layers neural network. The neural net has a typical bottleneck structure. The first (*input*) and the last (*output*) layer have the same number of neurons, while the remaining hidden layers have less neuron than the first and the last ones. The second, the third and the fourth layer are called respectively *mapping*, *bottleneck* and *demapping* layer. Mapping and demapping layers have usually the same number of neurons. The number of the neurons of the bottleneck layer provides an ID estimate. The targets used to train Nonlinear PCA are simply the input vector themselves. Though autoassociative neural networks (ANNs) outperforms linear PCA, as ID estimators, in some contexts, ANNs present some drawbacks. ANNs cannot model curves or surfaces that intersect themselves. Moreover, ANN projections onto curves and surfaces are suboptimal [37].

2.2 Fractal-Based Methods

Fractal-based techniques are global methods that have been successfully applied to estimate the attractor dimension of the underlying dynamic system generating time series [27]. Unless other global methods, they can provide as ID estimation a non-integer value. Since fractals are generally[2] characterized by a non-integer dimensionality, for instance the dimension of Cantor's set and Koch's curve [38] is respectively $\frac{\ln 2}{\ln 3}$ and $\frac{\ln 4}{\ln 3}$, these methods are called *fractal*. In nonlinear dynamics many definitions of *fractal* dimensions [13] have been proposed. The *Box-Counting* and the *Correlation* dimension are the most popular. The first definition of dimension (*Hausdorff dimension*) [13,41] is due to Hausdorff [19]. Since the Hausdorff dimension is not easy to evaluate, in practical application it is replaced by an upper bound that differs only in some constructed examples: the *Box-Counting dimension* (or *Kolmogorov capacity*) [41].

Kégl's Algorithm. Let $\Omega = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_\ell\}$ be a set of points in $\mathbb{R}^n$ of cardinality ℓ. We denote with $\nu(r)$ the number of the boxes (i.e., hypercubes) of size r required to cover Ω. It can be proven [41] that $\nu(r)$ is proportional to $(\frac{1}{r})^d$, where d is the *dimension* of the set Ω. This motivates the following definition. The Box-Counting dimension (or *Kolmogorov capacity*) D_B of the set Ω [41] is defined by

$$D_B = \lim_{r \to 0} \frac{\ln(\nu(r))}{\ln(\frac{1}{r})} \tag{1}$$

where the limit is assumed to exist. Recently Kégl [29], has proposed a fast algorithm (*Kégl's algorithm*) to estimate the Box-Counting dimension. Kégl's algorithm is based on the observation that $\nu(r)$ is equivalent to the cardinality of the maximum independent vertex set $MI(G_r)$ of the graph $G_r(V, E)$ with vertex set $V = \Omega$ and edge set $E = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \,|\, d(\boldsymbol{x}_i, \boldsymbol{x}_j) < r\}$. Kégl has proposed to estimate $MI(G)$ using the following greedy approximation. Given a data set Ω, we start with an empty set $\mathcal{C}$. In an iteration over Ω, we add to $\mathcal{C}$ data points that are at distance of at least r from all elements of $\mathcal{C}$. The cardinality of $\mathcal{C}$, after every point in Ω has been visited, is the estimate of $\nu(r)$. The Box-Counting dimension estimate is given by:

$$D_B = -\frac{\ln \nu(r_2) - \ln \nu(r_1)}{\ln r_2 - \ln r_1} \tag{2}$$

where r_2 and r_1 are values that can be set up heuristically. It can be proven [29] that the complexity of Kegl's algorithm is given by $O(D_B \ell^2)$, where ℓ and D_B are the cardinality and the dimensionality of the data set, respectively.

Grassberger-Procaccia Algorithm. A good substitute for the Box-Counting dimension can be the *Correlation dimension* [18]. Due to its computational simplicity, the Correlation dimension is successfully used to estimate the dimension

[2] Fractals have not always non-integer dimensionality. For instance, the dimension of *Peano's curve* is *2*.

of attractors of dynamical systems. The *Correlation dimension* [18] of a set Ω is defined as follows. If the *correlation integral* $C_m(r)$ is defined as:

$$C_m(r) = \lim_{\ell \to \infty} \frac{2}{\ell(\ell-1)} \sum_{i=1}^{\ell} \sum_{j=i+1}^{\ell} I(\|\boldsymbol{x}_j - \boldsymbol{x}_i\| \leq r) \tag{3}$$

where I is an *indicator function*[3], then the Correlation dimension D of Ω is:

$$D = \lim_{r \to 0} \frac{\ln(C_m(r))}{\ln(r)} \tag{4}$$

It can be proven that the Correlation Dimension is a lower bound of the Box-Counting Dimension. The most popular method to estimate Correlation dimension is the *Grassberger-Procaccia algorithm* [18]. This method consists in plotting $\ln(C_m(r))$ versus $\ln(r)$. The Correlation dimension is the slope of the linear part of the curve (see Fig. 2b). The computational complexity of the Grassberger-Procaccia algorithm is $O(\ell^2 s)$ where ℓ is the cardinality of the data set and s is the number of different times that the integral correlation is evaluated, respectively. However, there are efficient implementations of the Grassberger-Procaccia algorithm whose complexity does not depend on s. For these implementations, the computational complexity is $O(\ell^2)$.

Takens' Method. Takens [50] has proposed a method, based on *Fisher's method of Maximum Likelihood* [12], that allows to estimate the correlation dimension with a standard error. Let Q be the following set $Q = \{q_k \mid q_k < r\}$ where q_k is the the Euclidean distance between a generic couple of points of Ω and r (*cut-off radius*) is a real positive number. Using the Maximum Likelihood principle it can prove that the expectation value of the Correlation Dimension $\langle D_c \rangle$ is:

$$\langle D_c \rangle = -\left(\frac{1}{|Q|} \sum_{k=1}^{|Q|} q_k \right)^{-1} \tag{5}$$

where $|Q|$ stands for the cardinality of Q. Takens' method presents some drawbacks. It requires some heuristics to set the radius [53]. Besides, the method is optimal only if the correlation integral $C_m(r)$ assumes the form $C_m(r) = ar^D[1 + br^2 + o(r^2)]$ where a and b are constants, otherwise it can perform poorly [52]. Finally, Hein and Audibert [20] proposed a generalization of the correlation integral, in term of U-statistics [22], defined as follows:

$$U_{n,h}(K) = \frac{2}{\ell(\ell-1)} \sum_{i=1}^{\ell} \sum_{j=i+1}^{\ell} \frac{1}{h^m} K(\|\boldsymbol{x}_j - \boldsymbol{x}_i\|^2 / h^2) \tag{6}$$

where $K(\cdot)$ is a generic kernel of band width h and m is the dimensionality of the manifold where the data are assumed that lie. On the basis of the Hoeffding

[3] $I(\lambda)$ is 1 iff condition λ holds, 0 otherwise.

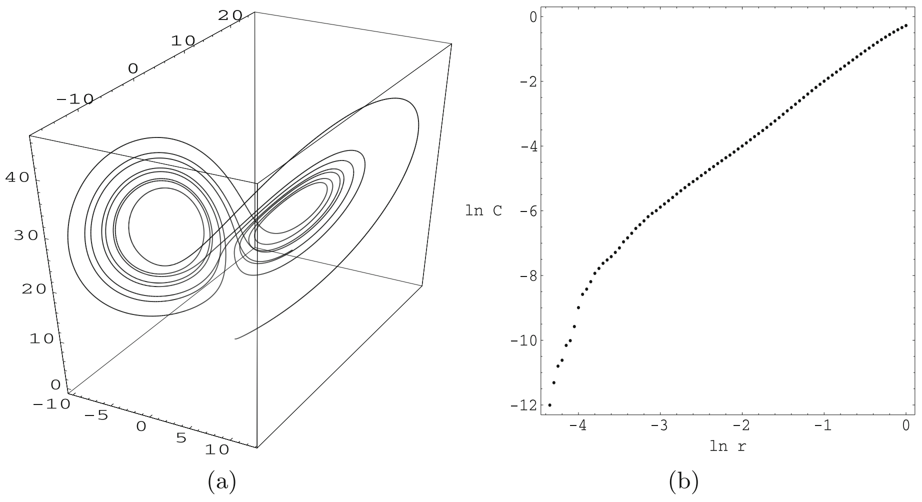


Fig. 2. (a) The attractor of the Lorentz system. (b) The log-log plot on Data set A. Data set A is a real data time series generated by a Lorentz-like system, implemented by NH$_3$-FIR lasers.

Theorem [23], to guarantee the convergence of the U-statistics the bandwidth h must fulfill $\ell h^m \to \infty$. Hein and Audibert used this property by fixing aconvergence rate for each dimension, that means weare fixing h as a function of the data set cardinality ℓ. Then the Eq. (6) is computed for subsamples of different cardinalities, where h varies according to the function we have fixed. ID is determined by the U-statistic which has the smallest slope as a function of h. It is worth to remark that, Hein and Audibert's algorithm tries, even if partially, to address the problem of ID dependence on the data scale.

Limitations of Fractal Methods. In addition to the drawbacks previously exposed, estimation methods based on fractal techniques have a fundamental limitation. It has been proved [14] that in order to get an accurate estimate of the dimension D, the set cardinality ℓ has to satisfy the so-called *Eckmann-Ruelle's inequality*, $D < 2\log_{10}\ell$.

The inequality shows that the number ℓ of data points required to accurately estimate the dimension of a D-dimensional set is at least $10^{\frac{D}{2}}$. Even for low dimensional sets this leads to huge values of ℓ. In order to cope with this problem and to improve the reliability of the measure for low values of ℓ, the *method of surrogate data* [54] has been proposed. The method of surrogate data is an application of *bootstrap* [15]. Given a data set Ω, the method of surrogate data consists in creating a new synthetic data set Ω', with larger cardinality, that has the same statistical properties of Ω, namely the same mean, variance and Fourier Spectrum. Although the cardinality of Ω' can be chosen arbitrarily, the method of surrogate data is infeasible when the dimensionality of the data set is high.

In-fact a 18-dimensional data set to be estimated must have at least, on the base of the Eckmann-Ruelle' inequality, 10^9 points. Camastra and Vinciarelli [7,8] proposed a procedure to power Grassberger and Procaccia method (GP method), establishing empirically how much GP method underestimates the dimensionality of a data set when data set cardinality is unadequate. Consider a set Ω of cardinality ℓ. The procedure is the following:

1. Create a set Ω', whose ID d is known, with the same cardinality ℓ of Ω. For instance, Ω' could be composed of ℓ data points randomly generated in a d-dimensional hypercube.
2. Measure the correlation dimension D of Ω' with the GP method.
3. Repeat the two previous steps for T different values of d, obtaining the set $C = \{(d_i, D_i) : i = 1, 2, \ldots, T)$.
4. Perform a best-fitting to the data points in C. A plot (*reference curve*) Γ of D versus d is generated. The reference curve allows to infer the value of D when d is known.
5. The correlation dimension D of Ω is computed by GP method and, using Γ, the intrinsic dimension of Ω can be estimated.

The procedure assumes implicitly that the curve Γ depends on ℓ and the dependence of Γ on the Ω' sets are negligible. It is worth to mention that Oganov and Valle [56] used GP method in conjunction to Camastra and Vinciarelli procedure's to estimate ID of Crystal Fingerprint spaces.

2.3 Multidimensional Scaling and Other Methods

Multidimensional Scaling (MDS) [44] methods are projection techniques that tend to preserve, as much as possible, the distances among data. Therefore data that are close in the original data set should be projected in such a way that their projections, in the new space (*output space*), are still close. To each projection is associated an index, usually defined *stress*, that measures the goodness of the projection. The best projection is the one whose stress is minimal. Examples of the Multidimensional scaling methods are Bennett's algorithm [3], that now has only historical interest, *MDSCAL* [31], *Sammon's mapping* [47]. In the Other Methods category, are collected the methods that do not belong to fractal, projection and MDS categories. To Other Methods category belong Costa-Hero [11] algorithm and the algorithms recently proposed by Rozza et al. [46] and Lombardi et al. [35]. For the sake of brevity, we only describe the first algorithm. Costa-Hero's algorithm assumes that data lie on a manifold. The algorithm exploits entropic graphs on in order to estimate the ID dimensionality and the entropy of the manifold. The algorithm is founded on the fact that the length function, computed on the whole graph, depends on ID.

3 Local Methods

Local methods are algorithms that provide an ID estimation using the information contained in sample neighborhoods, avoiding the projection of the data onto a lower-dimensional manifold. In this case, data do not lie on a unique manifold of constant dimensionality but on multiple manifolds of different dimensionalities. Since a unique ID estimate for the whole data is clearly not meaningful, it prefers to provide an ID estimate for each small subset of data, assuming that it lies on a manifold of constant dimensionality. More formally, local (or *topological*) methods try to estimate the topological dimension of the data manifold. The definition of topological dimension was given by Brouwer [21] in 1913. Topological dimension is the basis dimension of the local linear approximation of the hypersurface where the data reside, i.e., the tangent space. For example, if the data set lies on an m-dimensional submanifold, then it has an m-dimensional tangent space at every point in the set. For instance, a sphere has a two-dimensional tangent space at every point and may be viewed as a two-dimensional manifold. Since the ID of the sphere is three, the topological dimension represents a lower bound of ID. If the data does not lie on a manifold, the definition of topological dimension does not directly apply. Sometimes the topological dimension is also referred to simply as the *local dimension*. This is the reason why the methods that estimate the topological dimension are called local. Algorithms that belong to this category are Fukunaga-Olsen [17], Bruske-Sommer [5], Trunk [55], Pettis et al. [42] and Verveer and Duin [58] ones.

3.1 Fukunaga-Olsen's Algorithm

Fukunaga-Olsen's algorithm is based on the observation that for data embedded in a linear subspace, the dimension is equal to the number of non-zero eigenvalues of the covariance matrix. Besides, Fukunaga and Olsen assume that the intrinsic dimensionality of a data set can be computed by dividing the data set in small regions (*Voronoi tesselation* of data space). Voronoi tesselation can be performed by means of a clustering algorithm, e.g., LBG [33]. In each region (*Voronoi set*) the surface in which the vectors lie is approximately linear and the eigenvalues of the local covariance matrix are computed. Eigenvalues are normalized by dividing them by the largest eigenvalue. The intrinsic dimensionality is defined as the number of normalized eigenvalues that are larger than a threshold T. Although Fukunaga and Olsen proposed for T, on the basis of heuristic motivations, values such as 0.05 and 0.01, it is not possible to fix a threshold value T good for every problem.

3.2 TRN-Based and Local MDS Methods

Topology Representing Network (TRN) is a unsupervised neural network proposed by Martinetz and Schulten [39]. They proved that TRN are optimal topology preserving maps i.e., TRN preserves in the map the topology originally present in the data. Bruske and Sommer [5] proposed to improve Fukunaga-Olsen's algorithm using TRN in order to perform the Voronoi tesselation of the data space. In detail, the algorithm proposed by Bruske and Sommer is the following. An optimal topology preserving map G, by means of a TRN, is computed. Then, for each neuron $i \in G$, a PCA is performed on the set Q_i consisting of the differences between the neuron i and all of its m_i closest neurons in G. Bruske-Sommer's algorithm shares with Fukunaga-Olsen's one the same limitations: since none of the eigenvalues of the covariance matrix will be null due to noise, it is necessary to use heuristic thresholds in order to decide whether an eigenvalue is significant or not. Finally, we conclude the section on local methods quoting the local MDS methods. As the global MDS methods discussed in Sect. 2.3, local MDS methods are projection techniques that tend to preserve, as much as possible, the distances among data. In local MDS, in an analogous manner to global MDS, to each projection is associated an index or a cost that measures the goodness of the projection. Unlike MDS methods, where the whole data set is considered, local MDS methods work only on a small subset of data. Examples of Local MDS methods are ISOMAP [51] and Local Linear Embedding (LLE) [45]. The method for estimating ID is the same of global MDS. Compute several MDS projection considering different dimensionality for the output space. Pick the MDS projection with the best index or the minimum cost. The ID is given by the dimensionality of the output space of the MDS projection selected.

4 Mixed Methods

The most relevant methods that belong to this category are Levina-Bickel [32] and Carter-Raich-Hero algorithms [9]. For the sake of brevity, we only describe the former algorithm.

4.1 Levina-Bickel Algorithm

The Levina-Bickel algorithm provides a maximum likelihood ID estimate. The Levina-Bickel algorithm derives the maximum likelihood estimator (MLE) of the intrinsic dimensionality D from a data set $\Omega = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_\ell) \in \mathbb{R}^n$. The dataset Ω represents an embedding of a lower-dimensional sample, i.e., $\boldsymbol{x}_i = g(Y_i)$ where Y_i are sampled from an unknown smooth density f on $\mathbb{R}^D$ with $D \leq n$, g is a smooth mapping. Last assumption guarantees that close data in $\mathbb{R}^D$ are mapped to close neighbors in the embedding. That being said, we fix a data point $\boldsymbol{x} \in \mathbb{R}^n$ assuming that $f(\boldsymbol{x})$ is constant in a sphere $S_{\boldsymbol{x}}(r)$ centered in $\boldsymbol{x}$ of radius r and we view Ω as a homogeneous Poisson process in $S_{\boldsymbol{x}}(r)$. Given the inhomogeneous process $\{P(t, \boldsymbol{x}), 0 \leq t \leq r\}$

$$P(t, \boldsymbol{x}) = \sum_{i=1}^{\ell} I(\boldsymbol{x}_i \in S_{\boldsymbol{x}}(t)), \tag{7}$$

which counts the data whose distance from $\boldsymbol{x}$ is less than t. If we approximate it by means a Poisson process and we neglect the dependence on $\boldsymbol{x}$, the rate $\lambda(t)$ of the process $P(t)$ is given by:

$$\lambda(t) = f(\boldsymbol{x}) V(D) D t^{D-1}, \tag{8}$$

where $V(D)$ is the volume of a D-dimensional unit hypersphere. The Eq. (8) is justified by the Poisson process properties since the surface area of the sphere $S_{\boldsymbol{x}}(t)$ is $\frac{d}{dt}[V(D)t^D] = V(D)Dt^{D-1}$. If we define $\theta = log f(\boldsymbol{x})$, the log-likelihood of the process $P(t)$ [49] is:

$$L(D,\theta) = \int_0^r log\lambda(t)dP(t) - \int_0^r \lambda(t)dt. \tag{9}$$

The equation describes an exponential family for which a maximum likelihood estimator exists with probability that tends to 1 as the number of samples ℓ tends to infinity. The maximum likelihood estimator is unique and must satisfy the following equations:

$$\frac{\partial L}{\partial \theta} = \int_0^r dP(t) - \int_0^r \lambda(t)dt = P(r) - e^{\theta}V(D)r^D = 0. \tag{10}$$

$$\frac{\partial L}{\partial D} = \left(\frac{1}{D} + \frac{V'(D)}{V(D)}\right)P(r) + \int_0^r log\; t\; dP(t) + \\ - e^{\theta}V(D)r^D\left(log\; r + \frac{V'(D)}{V(D)}\right) = 0. \tag{11}$$

If we plug the Eq. (10) into the Eq. (11) we obtain the maximum likelihood estimate for the dimensionality D:

$$\hat{D}_r(\boldsymbol{x}) = \left[\frac{1}{P(r,\boldsymbol{x})}\sum_{j=1}^{P(r,\boldsymbol{x})} log\frac{r}{T_j(\boldsymbol{x})}\right]^{-1}, \tag{12}$$

where $T_j(\boldsymbol{x})$ denotes the Euclidean distance between $\boldsymbol{x}$ and its j-th nearest neighbor. Levina and Bickel suggest to fix the number of the neighbors k rather than the radius of the sphere r. Therefore the estimate becomes:

$$\hat{D}_k(\boldsymbol{x}) = \left[\frac{1}{k-1}\sum_{j=1}^{k-1} log\frac{T_k(\boldsymbol{x})}{T_j(\boldsymbol{x})}\right]^{-1}. \tag{13}$$

The estimate of the dimensionality is obtained averaging on all data points of the data set Ω, that is:

$$\hat{D}_k = \frac{1}{\ell}\sum_{i=1}^{\ell} \hat{D}_k(\boldsymbol{x}_i) \tag{14}$$

The estimate of the dimensionality depends on the value of k. Levina and Bickel suggest to average over a range of values of $k = k_1, \dots, k_2$ obtaining the final estimate of the dimensionality, i.e.,

$$\hat{D} = \frac{1}{k_2 - k_1 + 1}\sum_{k=k_1}^{k=k_2} \hat{D}_k. \tag{15}$$

David Mac Kay and Zoubin Ghamarani, in an unpublished comment [36], made a strong criticism against Levina and Bickel' s procedure of the global ID estimation. Instead, they proposed to average the inverse of the estimators $\hat{D}_k(\boldsymbol{x}_i)$. In this way, the Eq. (14) has to be replaced with:

$$\hat{D}_k = \frac{\ell(k-1)}{\displaystyle\sum_{i=1}^{\ell}\sum_{j=1}^{k-1} log \frac{T_k(\boldsymbol{x}_i)}{T_j(\boldsymbol{x}_i)}} \tag{16}$$

Using the same Levina and Bickel's approach, the final estimate of the dimensionality has to be obtained averaging $\hat{D}_k$ over a range of values of $k = k_1, \dots, k_2$ obtaining the final estimate of the dimensionality expressed by Eq. (15). Regarding the computational complexity, the Levina-Bickel algorithm requires a sorting algorithm[4], whose complexity is $O(\ell \log \ell)$, where ℓ denotes the cardinality of the data set. Hence the computational complexity for estimating $\hat{D}_k$ is $O(k\ell^2 \log \ell)$, where k denotes the numbers of the neighbors that have to be considered. Besides, Levina and Bickel suggest to consider an average estimate repeating the estimate D_k s times, where s is the difference between the maximum and the minimum value that k can assume, i.e., k_2 and k_1, respectively. Therefore the overall computational complexity of the Levina-Bickel algorithm is $O(k_2 s \ell^2 \log \ell)$.

5 ID Estimation Methods Benchmarking

A crucial issue in ID estimation is the experimental validation of the algorithms designed for ID estimation. The experimental validation of such an algorithm requires benchmarks, i.e., data sets, Benchmarks can be of two different types: synthetical or real data. Regarding synthetical benchmarks, it is not difficult to build synthetical data sets of given ID [20]. Moreover, the literature offer a certain number of synthetical benchmarks, both low-dimensional and high dimensional. To this purpose, it is worth to mention 2-dimensional *Swiss Roll* [51], 3-dimensional 10-Möbius strip [20], 9-dimensional data set D of Santa Fe time series competition [43], 12-dimensional manifold [20]. Unlike synthetical benchmarks, it can be cumbersome to get real data benchmarks of known ID. Firstly, it is necessary to split the benchmarks in two subfamilies: low-dimensional and high-dimensional. Regarding low-dimensional real data benchmarks, the literature offers a limited availability of benchmarks, e.g., the 3-dimensional Face Set [51] and the attractors in the phase space, of known dimensionality, generated, using *method of delays* [41], by real data time series. To this purpose, it is worth to mention the Lorentz attractor generated by the data set A[5] [24]

[4] The complexity of effective sorting algorithms (e.g., mergesort and heapsort) is $\ell \log \ell$, where ℓ is the number of elements that have to be sorted.

[5] The data set A is a real data time series generated by a Lorentz-like chaotic system, implemented by NH_3-FIR lasers.

Table 1. Chua's circuit and Data set A attractor dimension estimates by Kégl, Levina-Bickel, Grassberger-Procaccia methods.

	Data set A attractor dimension	Chua's circuit attractor dimension
Kégl estimate	2.02	2.14
Levina-Bickel estimate	2.35	2.26
Grassberger-Procaccia estimate	2.00	2.20
Theoretical value	2.06	$\sim$2.26

and Chua's attractor generated by a real data time series, measured from a hardware realization [1] of Chua's circuit [10]. In Table 1 some experimental comparisons [6] among ID estimators, performed on Data Set A and Chua's circuit, are reported. If we pass to consider high-dimensional real data benchmarks of known ID, the situation becomes very difficult. To our best knowledge, the only high-dimensional benchmarks are the *Crystal Fingerprint spaces* (or *Crystal Fingerspaces*) [40,56] recently proposed by Oganov and Valle in Crystallography with the aim of representing crystalline structures. Crystal Fingerprint spaces are spaces built starting by the real measured distances between atoms in the crystalline structure. The theoretical ID of a Crystal Fingerspace, based on crystal degree of freedoms, is 3N+3, where N is the number of the atoms in the crystalline unitary cell. Crystal Fingerspaces have been derived for several crystal structures, e.g., 39-dimensional H_2O (crystalline cell with 8 atoms) and 147-dimensional SiO_2 (crystalline cell with 48 atoms). Crystal Fingerspace data are available at http://mariovalle.name/CrystalFp/index.php/CrystalFpLib/Data.

6 Conclusions

In the paper we have reviewed the intrinsic dimension estimation methods underlining their advances. Nevertheless, some problem remain open. As remarked previously, intrinsic dimension depends on the scale of data. Although some ID estimation methods [20,34] tried to take in account, even if partially, of the data scale, a reliable multiscale ID estimator is not available, yet. The other open problems are related to the robustness of ID estimators w.r.t. the curse of dimensionality. About this topic, there are two issues that remain to be fully addressed. The former issue is the following. Each ID estimation method should provide a lower bound on the cardinality in order to guarantee an accurate ID estimation. To our best knowledge, this lower bound [14,48] is available only for Correlation Dimension estimation methods, e.g., Eckmann-Ruelle's inequality, whereas the other algorithms fully ignored the topic. The latter issue is the lack of the robustness of ID estimators w.r.t. high dimensionality. Although an empirical solution [8] was proposed, the construction of a robust ID estimators w.r.t. high dimensionality remains one of the challange of the research in machine learning.

Acknowledgements. Firstly, the author wish to thank Mario Valle for having made public Crystal Fingerspace datasets and commenting on the draft. The author thanks the anonymous referees for their useful remarks.

References

1. Aguirre, L., Rodrigues, G., Mendes, E.: Nonlinear identification and cluster analysis of chaotic attractors from a real implementation of chua's circuit. Int. J. Bifurcat. Chaos **6**(7), 1411–1423 (1997)
2. Bellman, R.: Adaptive Control Processes: A Guided Tour. Princeton University Press, Princeton (1961)
3. Bennett, R.S.: The intrinsic dimensionality of signal collections. IEEE Trans. Inf. Theory **15**, 517–525 (1969)
4. Bishop, C.: Neural Networks for Pattern Recognition. Cambridge University Press, Cambridge (1995)
5. Bruske, J., Sommer, G.: Intrinsic dimensionality estimation with optimally topology preserving maps. IEEE Trans. Pattern Anal. Mach. Intel. **20**(5), 572–575 (1998)
6. Camastra, F., Filippone, M.: A comparative evaluation of nonlinear dynamics methods for time series prediction. Neural Comput. Appl. **18**(8), 1021–1029 (2009)
7. Camastra, F., Vinciarelli, A.: Intrinsic dimension estimation of data: an approach based on grassberger-procaccia's algorithm. Neural Process. Lett. **14**(1), 27–34 (2001)
8. Camastra, F., Vinciarelli, A.: Estimating the intrinsic dimension of data with a fractal-based method. IEEE Trans. Pattern Anal. Mach. Intel. **24**(10), 1404–1407 (2002)
9. Carter, K., Raich, R., Hero, A.: On local intrinsic dimension estimation and its application. IEEE Trans. Sig. Process. **58**(2), 650–663 (2010)
10. Chua, L., Komuro, M., Matsumoto, T.: The double scroll. IEEE Trans. Circuits Syst. **32**(8), 797–818 (1985)
11. Costa, J., Hero, A.: Geodesic entropic graphs for dimension and entropy estimation in manifold learning. IEEE Trans. Sig. Process. **52**(8), 2210–2221 (2004)
12. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley, New York (2001)
13. Eckmann, J.P., Ruelle, D.: Ergodic theory of chaos and strange attractors. Rev. Mod. Phys. **57**, 617–659 (1985)
14. Eckmann, J.P., Ruelle, D.: Fundamental limitations for estimating dimensions and lyapounov exponents in dynamical systems. Physica **D–56**, 185–187 (1992)
15. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman and Hall, New York (1993)
16. Fukunaga, K.: Intrinsic dimensionality extraction. In: Krishnaiah, P.R., Kanal, L.N. (eds.) Classification, Pattern Recognition and Reduction of Dimensionality. Handbook of Statistics, pp. 347–360. North Holland, Amsterdam (1982)
17. Fukunaga, K., Olsen, D.: An algorithm for finding intrinsic dimensionality of data. IEEE Trans. Comput. **C–20**(2), 176–183 (1971)
18. Grassberger, P., Procaccia, I.: Measuring the strangeness of strange attractors. Physica D **9**, 189–208 (1983)
19. Hausdorff, F.: Dimension und äusseres mass. Mathematische Annalen **79**, 57 (1918)
20. Hein, M., Audibert, J.Y.: Intrinsic dimensionality estimation of submanifolds in $\mathbb{R}^d$. In: ICML 2005 Proceedings of 22^{nd} International Conference on Machine Learning, pp. 289–296 (2005)

21. Heyting, A., Freudenthal, H.: Collected Works of L.E.J Brouwer. North Holland Elsevier, Amsterdam (1975)
22. Hoeffding, W.: A class of statistics with asymptotically normal distributions. Ann. Stat. **19**, 293–325 (1948)
23. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**, 13–30 (1963)
24. Hubner, U., Weiss, C., Abraham, N., Tang, D.: Lorentz-like chaos in nh$_3$-fir lasers. In: Gershenfeld, N.A., Weigend, S.A. (eds.) Time Series Prediction: Forecasting the Future and Understanding the Past, pp. 73–104. Addison Wesley, Reading (1994)
25. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice Hall, New Jersey (1988)
26. Jollife, I.T.: Principal Component Analysis. Springer, New York (1986)
27. Kaplan, D., Glass, L.: Understanding Nonlinear Dynamics. Springer, New York (1995)
28. Karhunen, J., Joutsensalo, J.: Representations and separation of signals using nonlinear PCA type learning. Neural Netw. **7**(1), 113–127 (1994)
29. Kégl, B.: Intrinsic dimension estimation using packing numbers. In: Thrun, S., Saul, L.K., Schölkopf, B. (eds.) Advances in Neural Information Processing. MIT Press, Cambridge (2003)
30. Kirby, M.: Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns. Wiley, New York (2001)
31. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika **29**, 1–27 (1964)
32. Levina, E., Bickel, P.: Maximum likelihood estimation of intrinsic dimension. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing, pp. 777–784. MIT Press, Cambridge (2005)
33. Linde, Y., Buzo, A., Gray, R.: An algorithm for vector quantizer design. IEEE Trans. Commun. **28**(1), 84–95 (1980)
34. Little, A., Jung, Y.M., Maggioni, M.: Multiscale estimation of intrinsic dimensionality of a data set. In: Manifold Learning and Its Applications: Papers from the AAAI Fall Symposium, pp. 26–33. IEEE (2009)
35. Lombardi, G., Rozza, A., Ceruti, C., Casiraghi, E., Campadelli, P.: Minimum neighbor distance estimators of intrinsic dimension. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 374–389. Springer, Heidelberg (2011)
36. MacKay, D., Ghamarani, Z.: Comments on 'Maximum likelihood estimation of intrinsic dimension by E. Levina and M. Bickel', University of Cambridge (2005). http://inference.phy.cam.uc.uk/mackay/dimension
37. Malthouse, E.C.: Limitations of nonlinear PCA as performed with generic neural networks. IEEE Trans. Neural Netw. **9**(1), 165–173 (1998)
38. Mandelbrot, B.: Fractals: Form, Chance and Dimension. Freeman, San Francisco (1977)
39. Martinetz, T., Schulten, K.: Topology representing networks. Neural Netw. **3**, 507–522 (1994)
40. Oganov, A., Valle, M.: How to quantify energy landscapes of solids. J. Chem. Phys. **130**, 104504 (2009)
41. Ott, E.: Chaos in Dynamical Systems. Cambridge University Press, Cambridge (1988)
42. Pettis, K., Bailey, T., Jain, T., Dubes, R.: An intrinsic dimensionality estimator from near-neighbor information. IEEE Trans. Pattern Anal. Mach. Intel. **1**(1), 25–37 (1979)

43. Pineda, F., Sommerer, J.: Estimating generalized dimensions and choosing time delays: a fast algorithm. In: Weigend, S., Gershenfeld, N.A. (eds.) Time Series Prediction: Forecasting the Future and Understanding the Past, pp. 367–385. Addison Wesley, Reading (1994)
44. Romney, A.K., Shepard, R.N., Nerlove, S.B.: Multidimensionaling Scaling, vol. I. Theory. Seminar Press, New York (1972)
45. Roweis, S., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(12), 2323–2326 (2000)
46. Rozza, A., Lombardi, G., Rosa, M., Casiraghi, E., Campadelli, P.: IDEA: intrinsic dimension estimation algorithm. In: Maino, G., Foresti, G.L. (eds.) ICIAP 2011, Part I. LNCS, vol. 6978, pp. 433–442. Springer, Heidelberg (2011)
47. Sammon, J.W.J.: A nonlinear mapping for data structure analysis. IEEE Trans. Comput. **C–18**, 401–409 (1969)
48. Smith, R.: Optimal estimation of fractal dimension. In: Casdagli, M., Eubank, S. (eds.) Nonlinear Modeling and Forecasting, pp. 115–135. Addison Wesley, New York (1992)
49. Snyder, D.: Random Point Processes. Wiley, New York (1975)
50. Takens, F.: On the numerical determination of the dimension of an attractor. In: Dynamical Systems and Bifurcations, Proceedings Groningen 1984, pp. 99–106. Addison Wesley (1985)
51. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(12), 2319–2323 (2000)
52. Theiler, J.: Lacunarity in a best estimator of fractal dimension. Phys. Lett. A **133**, 195–200 (1988)
53. Theiler, J.: Statistical precision of dimension estimators. Phys. Rev. **A41**, 3038–3051 (1990)
54. Theiler, J., Eubank, S., Longtin, A., Galdrikian, B., Farmer, J.D.: Testing for nonlinearity in time series: the method for surrogate date. Physica D **58**, 77–94 (1992)
55. Trunk, G.V.: Statistical estimation of the intrinsic dimensionality of a noisy signal collection. IEEE Trans. Comput. **25**, 165–171 (1976)
56. Valle, M., Oganov, A.: Crystal fingerprint space- a novel paradigm for studying crystal-structure sets. Acta Crystallogr. Sect. A **A66**, 507–517 (2010)
57. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
58. Verveer, P.J., Duin, R.: An evaluation of intrinsic dimensionality estimators. IEEE Trans. Pattern Anal. Mach. Intel. **17**(1), 81–86 (1995)
59. Villmann, T., Claussen, J.C.: Magnification control in self-organizing maps and neural gas. Neural Comput. **18**(2), 446–469 (2000)

A Novel Intrinsic Dimensionality Estimator Based on Rank-Order Statistics

S. Bassis[1], A. Rozza[2](✉), C. Ceruti[1], G. Lombardi[1], E. Casiraghi[1], and P. Campadelli[1]

[1] Dipartimento di Informatica, Università degli Studi di Milano, via Comelico 39-41, 20135 Milano, Italy
{bassis,ceruti,lombardi,casiraghi,campadelli}@di.unimi.it

[2] Research Team, Hyera Software, Via Mattei 2, Coccaglio (BS), Italy
alessandro.rozza@hyera.com

Abstract. In the past two decades the estimation of the intrinsic dimensionality of a dataset has gained considerable importance, since it is a relevant information for several real life applications. Unfortunately, although a great deal of research effort has been devoted to the development of effective intrinsic dimensionality estimators, the problem is still open. For this reason, in this paper we propose a novel robust intrinsic dimensionality estimator that exploits the information conveyed by the normalized nearest neighbor distances, through a technique based on rank-order statistics that limits common underestimation issues related to the edge effect. Experiments performed on both synthetic and real datasets highlight the robustness and the effectiveness of the proposed algorithm when compared to state-of-the-art methodologies.

Keywords: Intrinsic dimensionality estimation · Manifold learning · Rank-order statistics

1 Introduction

In the last decade a great deal of research work has been devoted to the development of the intrinsic dimensionality (`id`) estimators. To this aim, considering a dataset $\boldsymbol{X}_N \equiv \{\boldsymbol{x}_i\}_{i=1}^N \subset \Re^D$, the feature vectors $\boldsymbol{x}_i$ are generally viewed as points constrained to lie on a low dimensional manifold $\boldsymbol{\mathcal{M}} \subseteq \Re^d$ embedded in a higher dimensional space $\Re^D$, where d is the intrinsic dimensionality to be estimated. In more general terms, according to [11], $\boldsymbol{X}_N$ is said to have `id` equal to $d \in \{1..D\}$ if its elements lie entirely within a d-dimensional (locally) smooth manifold embedded in $\Re^D$.

The knowledge of dataset's `id` is crucial for several applications in the field of Artificial Intelligence, as it is highlighted by the following reasons. At first, dimensionality reduction techniques, which are often used to reduce the "curse of dimensionality" effect [17] by computing a more compact representation of the data, are profitable when the number of projection dimensions is the minimal

F. Masulli et al. (Eds.): CHDD 2012, LNCS 7627, pp. 102–117, 2015.
DOI: 10.1007/978-3-662-48577-4_7

one that allows to retain the maximum amount of useful information expressed by the data. Furthermore, when using an auto-associative neural network [19] to perform a nonlinear feature extraction, the id can suggest a reasonable value for the number of hidden neurons. Moreover, according to the statistical learning theory [32], the capacity and the generalization capability of a classifier may depend on the id. In particular, in [10] the authors mark that, in order to balance a classifier's generalization ability and its empirical error, the complexity of the classification model should also be related to the id of the available dataset. Finally, as it has been recently shown in [2], id estimation methods are used to evaluate the model order in a time series, which is crucial to make reliable time series predictions. This consideration is supported by the fact that the domain of attraction of a nonlinear dynamic system has a very complex geometric structure and the studies on the geometry of the attraction domain are closely related to fractal geometry, and therefore to fractal dimension.

Unfortunately, although a great deal of research effort has been focused on the development of id estimators, the problem is still open. For this reason, in this paper we present a novel methodology for intrinsic dimensionality estimation that exploits the information conveyed by the normalized neighbor distances through an efficient technique based on rank-order statistics. This method limits common underestimation issues related to the edge effect without introducing de-biasing procedures that reduce the effective sample size by filtering the border points [4].

This paper is organized as follows: Sect. 2 briefly surveys the most notable state-of-the-art techniques aimed at id estimation; in Sect. 3 base theoretical results laying foundations of the proposed estimator are presented; Sect. 4 describes the proposed algorithm, also providing a concise analysis of its properties; a detailed comparison with well-known methodologies on a wide family of datasets is reported in Sect. 5; finally, Sect. 6 reports conclusions and future works.

2 Related Works

In this section well-known id estimators are shortly recalled, highlighting their advantages and drawbacks.

In the following, the surveyed id estimators are grouped according to the categorization proposed in [22]. In particular, two groups have been identified by the authors: projection methods, which search for the best subspace where to project the data, and geometric id estimators, which exploit statistics related to either the distances between neighboring points or the fractal dimension, expressing them as functions of the id of the embedded manifold.

The most cited projection method is the Principal Component Analysis (PCA) [18], which projects the input dataset on the d directions of its maximum variance (principal components, PCs). More precisely, given a dataset of observed points $\boldsymbol{p}_t \in \Re^D$, PCA computes their low dimensional representations $\boldsymbol{p}_x \in \Re^d$ by projecting the points $\boldsymbol{p}_t \in \Re^D$ on the d directions (also called principal components, PCs) of their maximum variance. To this aim, PCA computes the $D \times d$ projection matrix $\boldsymbol{W}$ (whose columns $\boldsymbol{w}_i,\ i = 1, .., d$ are the PCs), so that the reduced points $\boldsymbol{p}_x$ are obtained from the observed points as $\boldsymbol{p}_x = \boldsymbol{W}^T (\boldsymbol{p}_t - \bar{\boldsymbol{p}}_t)$ (being $\bar{\boldsymbol{p}}_t$ the mean of the observed points), while, given the reduced data, the reconstructed points are $\tilde{\boldsymbol{p}}_t = \boldsymbol{W}\boldsymbol{p}_x + \bar{\boldsymbol{p}}_t$. The projection matrix $\boldsymbol{W}$ is the one that minimizes the sum of square distances between the observed points $\boldsymbol{p}_t$ and the reconstructed ones $\tilde{\boldsymbol{p}}_t$. According to [18], $\boldsymbol{W}$ can be obtained by selecting as PCs the eigenvectors corresponding to the highest eigenvalues of the data covariance matrix. Exploiting PCA, the intrinsic dimension d can be estimated by counting the number of retained PCs, that generally are the PCs whose corresponding (normalized) eigenvalue is higher than a threshold parameter. The problem of using PCA for id estimation relies in the difficulty of choosing a proper value for the threshold; moreover, PCA is a linear technique that cannot successfully deal with points drawn from nonlinearly embedded manifolds, and generally produces overestimations.

More accurate results can be obtained by applying a local PCA [12] that combines local id estimates computed in small subregions of the dataset; unfortunately, complications arise in the identification of local regions and in the thresholds selection [33].

Based on the observation that PCA and its variants are deterministic models lacking an associated probabilistic model for the observed data and a method for selecting the number of PCs to be retained (the id), in [30] the authors presented the Probabilistic PCA (PPCA). This approach reformulates PCA as the maximum likelihood solution of a specific latent variable model by considering a $d-$dimensional latent variable $\boldsymbol{x}$ (representing the reduced data) and setting their prior distribution to be a zero mean Gaussian whose covariance matrix is a $d-$dimensional identity matrix (that is $\boldsymbol{\mathcal{N}}(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{I}_d)$). The $D-$dimensional observed variable $\boldsymbol{t}$ is then defined as: $\boldsymbol{t} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$, that is a linear transformation of the latent variable $\boldsymbol{x}$ where $\boldsymbol{W}$ is a $D \times d$ parameter representing the projection matrix, $\boldsymbol{\mu}$ is a $D-$dimensional vector, and $\boldsymbol{\epsilon}$ is a zero-mean Gaussian distributed vector with covariance $\sigma^2 \boldsymbol{I}_D$ representing noise. Hence, the marginal distribution of the observed variable $\boldsymbol{t}$, is a constrained Gaussian distribution governed by the three parameters $\boldsymbol{W}$, $\boldsymbol{\mu}$, and σ. The maximum likelihood solution for these parameters allows to project the observed dataset on the reduced $d-$dimensional space and therefore represents the solution computed by means of PPCA.

Even though PPCA has obtained promising results, it still does not provide any mechanism for estimating the best value of the latent space dimensionality d, that is the id. For this reason, in [1] the author further extends the PPCA model by defining a Bayesian treatment of PCA (called Bayesian PCA or BPCA). To this aim a prior distribution over the three parameters $\boldsymbol{W}$, $\boldsymbol{\mu}$, and σ is introduced to formulate the posterior over the dataset and the predictive density by marginalizing over the three parameters. To automatically determine an effective dimensionality d for the latent variable $\boldsymbol{x}$, the author further introduces a "hierarchical" prior $p(\boldsymbol{W}|\boldsymbol{\alpha})$ over the parameter $\boldsymbol{W}$ that is governed by a $q-$dimensional vector of hyper-parameters $\boldsymbol{\alpha} = \alpha_1, .., \alpha_q$, where α_i controls the inverse "relevance" of $\boldsymbol{w}_i$, and q is initially set to $q = D - 1$. To make use of this model the author exploits a local Gaussian approximation to estimate the posterior distribution of $\boldsymbol{W}$, which must be marginalized to solve the problem.

Combining this procedure with maximum likelihood to determine the values of the α_i, the authors note that the vectors $\boldsymbol{w}_i$ for which there is insufficient support from the data will be driven to zero, with the corresponding $\alpha_i \longrightarrow \infty$, so that unused dimensions are switched off completely. The `id` is then defined as the number of `PC`s whose "relevance" value remains non-zero. This technique has been extended in [23] to cope with exponential family distributions, but this method requires the knowledge of the distribution underlying the data.

To achieve an automatic selection of meaningful `PC`s, in [14] the authors propose the Sparse Probabilistic Principal Component Analysis (`SPPCA`) that exploits the sparsity of the projection matrix through a probabilistic Bayesian formulation.

Unfortunately, all the above mentioned projection approaches cannot provide reliable `id` estimates since they are too sensitive to noise and parameter settings [22].

Perhaps, the most popular geometric `id` estimator is the Correlation Dimension (`CD`) [13] that is based on the assumption that the volume of a d-dimensional set scales as r^d with its size r. Since the performances of `CD` are affected by the choice of the scale r, in [15] the author suggests an estimator based on the asymptotes of a smoothed version of the `CD` estimate.

Another interesting technique, based on the analysis of point neighborhoods, is the Maximum Likelihood Estimator (`MLE`, [22]) that applies the principle of maximum likelihood to the distances between close neighbors, and derives the estimator by a Poisson process approximation. More precisely, calling k the number of neighbors, $\boldsymbol{x}_i$ the i^{th} point, and $T_k(\boldsymbol{x}_i)$ the radius of the smallest sphere centered in $\boldsymbol{x}_i$ containing exactly k neighbors, the local intrinsic dimension is estimated as:

$$\hat{d}(\boldsymbol{x}_i) = \left(\frac{1}{k} \sum_{j=1}^{k} \log \frac{T_{k+1}(\boldsymbol{x}_i)}{T_j(\boldsymbol{x}_i)} \right)^{-1}$$

In [8] the authors propose an algorithm, which exploits entropic graphs to estimate both the `id` and the intrinsic entropy of a manifold. This technique is based on the observation that the length function of such graphs, that is the sum of arc weights on the minimal graph that spans all the points in the dataset, is strongly dependent on d. The authors test their method by adopting either the geodesic minimal spanning tree (`GMST` [7]), where the arc weights are the geodetic distances computed through the `ISOMAP` [29] algorithm, or the `kNN`-graph (`kNNG` [8]), where the arc weights are based on the Euclidean distances, thus requiring a lower computational cost.

We note that the recalled neighborhood based estimators underestimate the `id` when its value is sufficiently high and, to our knowledge, only few works propose possible solutions to this problem [3,5,24,27,28].

3 Theoretical Results

Consider a manifold $\boldsymbol{\mathcal{M}} \equiv \Re^d$ embedded in a higher dimensional space $\Re^D$ through a locally isometric nonlinear smooth map $\phi : \Re^d \rightarrow \Re^D$; to estimate the id of $\boldsymbol{\mathcal{M}}$ by means of points drawn from the embedded manifold through a smooth probability density function (pdf) f, we need to identify a "mathematical object" depending only on d, and we should define a consistent estimator for d based on it.

Assume by hypothesis that the employed manifold sampling process is driven by a smooth pdf f; moreover, consider a spherical neighborhood of the origin $\mathbf{0}_d$ having radius ϵ; denoting with $\chi_{\boldsymbol{\mathcal{B}}_d(\mathbf{0}_d,1)}$ the indicator function on the unit ball $\boldsymbol{\mathcal{B}}_d(\mathbf{0}_d,1)$, the pdf restricted to such a neighborhood is:

$$f_\epsilon(\boldsymbol{z}) = \frac{f(\epsilon\boldsymbol{z})\chi_{\boldsymbol{\mathcal{B}}_d(\mathbf{0}_d,1)}(\boldsymbol{z})}{\int_{\boldsymbol{t}\in\boldsymbol{\mathcal{B}}_d(\mathbf{0}_d,1)} f(\epsilon\boldsymbol{t})d\boldsymbol{t}} \tag{1}$$

In [24] the authors prove the following:

Theorem 1. *Given $\{\epsilon_i\} \to 0^+$, Eq. (1) describes a sequence of pdfs having the unit d-dimensional ball as support; such sequence converges uniformly to the uniform distribution $\mathbf{B}_d$ in the ball $\boldsymbol{\mathcal{B}}_d(\mathbf{0}_d,1)$.*

Theorem 1 ensures that, from a theoretical standpoint, in our setting it is possible to assume uniformly distributed points in every neighborhood of $\boldsymbol{\mathcal{M}}$; in other words, we are allowed to define consistent estimators based on local information, assuming without loss of generality that the normalized points are uniformly drawn from $\boldsymbol{\mathcal{B}}_d(\mathbf{0}_d,1)$.

Our technique exploits the statistical properties of norms computed on points drawn from uniformly sampled hyperspheres to define a consistent estimator of the manifold's id. In particular, let $\{\boldsymbol{z}_i\}_{i=1}^k$ be a set of points[1] uniformly drawn from $\boldsymbol{\mathcal{B}}_d(\mathbf{0}_d,1)$. Let $R_i \in [0,1]$ denote the distance of $\boldsymbol{Z}_i$ from the center of the hypersphere $\mathbf{0}_d$, i.e. $R_i = ||\boldsymbol{Z}_i||$, where $||\cdot||$ is the L_2 norm operator. In [24] it is shown that the cumulative distribution function (cdf) F_{R_i} of R_i can be evaluated by means of the ratio between V_r that is the volume of a d dimensional hypersphere of radius r, and the analogously defined V_1, thus obtaining $F_{R_i}(r) = r^d$. Denoting with $\{R_{(i)}\}_{i=1}^k$ the order statistic of the observed distances, they follow a Beta distribution[2] with parameters i and $k-i+1$ ($\mathsf{Beta}\,(i, k-i+1)$) evaluated in r^d. This result, which is grounded on the theory of order statistics [34], arises from the fact that $R_{(i)} \le r$ iff $\#\{R_j \le r\}_{j=1}^k \ge i$, where $\#$ is the cardinality operator. To exploit useful asymptotic properties of Beta random variables (rvs), we opt to concentrate on the distribution of $R_{(k)}$ which, after some algebra, simplifies in a $\mathsf{Beta}\,(\tau,1)$, i.e. $F_{R_{(k)}} = r^\tau$, where $\tau = kd$.

To keep the underestimation effect [27] low when the number of border points increases in high dimensional spaces ("edge effect", [33]), we subtract from $R_{(k)}$ another rv suffering from the same problem and having the same behavior (for technical details, see Sect. 4 where the algorithm is described).

[1] By default, capital letters (such as U, X) will denote random variables (rv) and small letters (u, x) their corresponding realization.

[2] The pdf of a random variable X following a Beta distribution with parameters α and β is defined as $f_X(x|\alpha,\beta) = (\mathsf{B}(\alpha,\beta))^{-1}x^{\alpha-1}(1-x)^{\beta-1}$ where B is the Beta function providing for the normalization factor.

More precisely, let $\{\boldsymbol{z}'_i\}_{i=1}^{k'}$ be a further sample of points uniformly drawn from $\boldsymbol{\mathcal{B}}_d(\mathbf{0}_d, 1)$ and, with the same notation used so far, define R'_i and $R'_{(k')}$ to be respectively the norm of $\boldsymbol{z}'_i$, and the norm of the farthest point from the origin. The reproducibility property still holds for $R'_{(k')}$ that is distributed as a $\mathsf{Beta}\,(\tau', 1)$, where $\tau' = k'd$.

Before describing that the distribution law of the difference $\Delta_{(k,k')}$ between $R_{(k)}$ and $R'_{(k')}$ converges to the `cdf` F_L of an asymmetric Laplace distribution, let us introduce the following lemma.

Lemma 1. *Let B_a be a **rv** following a Beta distribution of parameters $(1, a)$ and Y_1 a **rv** following an Exponential distribution*[3] *of parameter $\lambda = 1$ ($\mathsf{Exp}\,(1)$). The sequence of **rvs** $\{aB_a\}_{a\in\mathbb{N}}$ converges to Y_1 both in mean (i.e. $aB_a \xrightarrow{L^2} Y_1$, or equivalently $\lim_{a\to\infty} aB_a = Y_1$) with order $\mathcal{O}(a^{-2})$ and uniformly (i.e. $\lim_{a\to\infty}\sup_{y\in\Re^+}|f_{aB_a}(y) - f_{Y_1}(y)| = 0$) with order $\mathcal{O}(a^{-1})$, where f_{aB_a} and f_{Y_1} are the **pdfs** of aB_a and Y_1, respectively.*

Proof. First of all, note that according to standard approximations for continuous univariate distributions [16], $\lim_{a\to\infty} aB_a = Y_1$. This is easily shown by noting that $F_{aB_a}(y) = \left(1 - \frac{y}{a}\right)^{a-1}$ converges to e^{-y} for $a \to \infty$.

Moreover, let us compute the mean square error as follows:

$$||aB_a - Y_1||_2^2 = \int_{y=0}^{a} \left(f_{aB_a}(y) - f_{Y_1}(y)\right)^2 + \int_{y=a}^{\infty} \left(f_{Y_1}(y)\right)^2 =$$

$$\frac{1}{4a-2} + 2a\mathrm{e}^{-a}(-a)^{-a}(\Gamma(a,-a) - \Gamma(a)) + 1 \quad (2)$$

where $\Gamma(a)$ and $\Gamma(a,z)$ are respectively the Gamma function ($\int_0^\infty t^{a-1}e^{-t}\,\mathrm{d}t$) and the Incomplete Gamma function ($\int_z^\infty t^{a-1}\mathrm{e}^{-t}\mathrm{d}t$). The log-log scale representation of Eq. (2) leads us to fit it through the power-law function $\phi_2(a) = \frac{0.065}{a^2}$ whose quality can be appreciated in Fig. 1(a). A similar investigation of $||aB_a - Y_1||_\infty = \max\left\{\max_{y\in[0,a]}\ |f_{aB_a}(y) - f_{Y_1}(y)|\,, f_{Y_1}(a)\right\}$ leads to its approximation through $\phi_\infty(a) = \frac{0.242}{a}$ (see Fig. 1(b)).

We are now ready to state the following theorem.

Theorem 2. *Let B_1 and B_2 be two Beta distributed **rv** of parameters $(a_1, 1)$ and $(a_2, 1)$. For $a_1, a_2 > M$ ($M \in \Re^+$ arbitrary large) the difference $\Delta = B_1 - B_2$ follows an asymmetric Laplace distribution [20] with parameters $\gamma = \sqrt{a_2/a_1}$ and $\sigma = \sqrt{2/a_1a_2}$.*

Proof. At first, we note that $\overline{B}_1 = 1 - B_1$ follows a $\mathsf{Beta}\,(1, a_1)$; moreover, according to Lemma 1, for large enough values of a we may approximate the distribution of $a_1\overline{B}_1$ with $\mathsf{Exp}\,(1)$. In particular, $\overline{B}_1 \sim 1/a_1 \mathsf{Exp}\,(1) = \mathsf{Exp}\,(a_1)$. After having applied the same reasoning to B_2, consider the difference Δ between B_1 and B_2:

$$\Delta = B_1 - B_2 = (1 - B_2) - (1 - B_1) = \overline{B}_2 - \overline{B}_1 \quad (3)$$

[3] The pdf of a random variable Y following an Exponential distribution is defined as $f_Y(y|\lambda) = \lambda e^{-\lambda y}$.

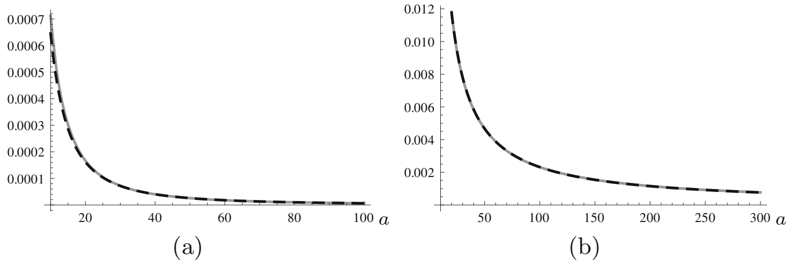


Fig. 1. Comparison between: (a) $||aB_a - Y_1||_2^2$ (dashed black curve) and its approximation ϕ_2 (gray curve), and (b) $||aB_a - Y_1||_\infty$ (dashed black curve) and its approximation ϕ_∞ (gray curve).

The distribution of the `rv` Δ *(reported in Eq. (3)) may be computed by evaluating the convolution integral between* $\mathsf{Exp}\,(a_2)$ *and* $\mathsf{Exp}\,(a_1)$*, thus obtaining:*

$$F_\Delta(\delta; a_1, a_2) = \begin{cases} \frac{a_2 e^{a_1 \delta}}{a_1 + a_2} & \text{if } \delta < 0 \\ 1 - \frac{a_1 e^{-a_2 \delta}}{a_1 + a_2} & \text{otherwise} \end{cases} \tag{4}$$

This looks like a special case of the well-known asymmetric Laplace distribution of Kozubowski and Podgórski [20], defined as:

$$F_L(l; \gamma, \sigma) = \begin{cases} \frac{\sqrt{2}\gamma \mathrm{e}^{\frac{\sqrt{2}y}{\gamma\sigma}}}{\sigma(\gamma^2+1)} & \text{if } \delta < 0 \\ \frac{\sqrt{2}\gamma \mathrm{e}^{-\frac{\sqrt{2}\gamma y}{\sigma}}}{\sigma(\gamma^2+1)} & \text{otherwise} \end{cases} \tag{5}$$

The connection between Eqs. (4) and (5) is attested by the substitutions $a_2 \leftarrow \frac{\sqrt{2}\gamma}{\sigma}$, $a_1 \leftarrow \frac{\sqrt{2}}{\gamma\sigma}$*, and their inverses* $\gamma \leftarrow \sqrt{\frac{a_2}{a_1}}$, $\sigma \leftarrow \sqrt{\frac{2}{a_1 a_2}}$.

As a direct consequence of Theorem 2, the difference $\Delta_{(k,k')}$ between $R_{(k)}$ and $R'_{(k')}$ may be well approximated by a `rv` following an asymmetric Laplace distribution of parameters $\gamma = \sqrt{\frac{\tau'}{\tau}}$ and $\sigma = \sqrt{\frac{2}{\tau\tau'}}$.

Remark 1. To numerically evaluate the feasibility of the approximations introduced in Lemma 1, consider firstly the relation that links the a_i parameters of the Beta distributions in Theorem 2 with the `id` d through the equality $a_i = dk$, with k neighborhood size. For instance, we need an `id` d of at least 25 to tolerate an absolute error $\epsilon = 0.001$: this value is obtained by solving $\phi_\infty(dk) = 0.001 = \frac{0.242}{kd}$ in d and assuming a neighborhood size k equal to 10 (as usually set in the literature and in our experiments). The needed `id` drops down to $d = 3$ when the tolerated error lowers to $\epsilon = 0.01$.

A more detailed investigation of the problem allows us to further improve the aforementioned relation. Looking at Fig. 2, we observe how the difference $f_{B_{a_i}}(y) - f_{Y_{a_i}}(y)$ between the `pdf` of a Beta distribution B_{a_i} with parameters $(1, a_i)$ and the `pdf` of an Exponential distribution Y_{a_i} of rate a_i tends to vanish

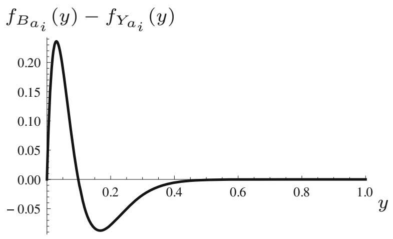


Fig. 2. Course of the difference $f_{B_{a_i}}(y) - f_{Y_{a_i}}(y)$ when $a_i = 20$.

for values of y close to 1. Moreover, thanks to the compression of norms, we know that the probability of observing a normalized distance under a given threshold (for example 0.8) between a point and its k-th neighbor in a `kNN` goes to zero as the dimensionality increases. In particular, denoting with B_{kd} a `rv` following a Beta distribution of parameters $(kd, 1)$, the value of the `cdf` evaluated in the chosen threshold, i.e. $F_{B_{kd}}(0.8) = 0.8^{dk}$, drops below 10^{-4} for $d \geq 4$ and $k = 10$.

This observation justifies the computation of $||aB_a - Y_1||_\infty$ only on that portion of the domain which effectively has a non negligible probability of being observed. In particular, the approximation:

$$\sup_{y \in [0.8a_i, \infty)} \left| f_{a_i B_{a_i}}(y) - f_{Y_1}(y) \right| \approx \frac{3 \times 10^{-15}}{y} \tag{6}$$

shows that the absolute error is so small to justify the use of the exponential approximation to the Beta distribution at least for $d \geq 4$.

As reported in [20], a Maximum Likelihood estimator (`ML`) could be found for the parameters γ and σ of F_L. In particular, denoting with $\boldsymbol{l} = \{l_1, \ldots, l_N\}$ a sample drawn from an asymmetric Laplace distribution, and using as statistics:

$$\alpha = \frac{1}{N} \sum_{j=1}^{N} \max\{l_j, 0\} \qquad \beta = -\frac{1}{N} \sum_{j=1}^{N} \min\{l_j, 0\} \tag{7}$$

a consistent `ML` for γ and σ is:

$$\hat{\gamma} = \sqrt[4]{\frac{\beta}{\alpha}} \qquad \hat{\sigma} = \sqrt{2}\sqrt[4]{\alpha}\sqrt[4]{\beta}\left(\sqrt{\alpha} + \sqrt{\beta}\right) \tag{8}$$

Finally, according to Theorem 2 and thanks to the properties of maximum likelihood estimators, a `ML` for the parameters of the $\mathsf{Beta}(\tau, 1)$ and $\mathsf{Beta}(\tau', 1)$ can be computed as:

$$\hat{\tau} = \frac{\sqrt{2}}{\hat{\gamma}\hat{\sigma}} \qquad \hat{\tau}' = \frac{\sqrt{2}\hat{\gamma}}{\hat{\sigma}} \tag{9}$$

Note that due to the usage of the `kNN` algorithm, a direct estimate of τ in $R_{(k)}$ may be poor. More precisely, in high dimensional spaces, the `kNN` method is strongly affected by the edge effect [33] that reduces the quality of the neighborhood estimation. The usage of $\Delta_{(k,k')}$ helps to reduce the aforementioned effects, as shown in our experimental results.

4 The Algorithm

In this section we show how the theoretical results presented in Sect. 3 can be exploited to estimate the `id` of a given dataset. In particular, the thorny problem is to understand how to retain the information contained in neighbor distances narrowing the underestimation caused by the edge effect, without wasting any sample data and still maintaining a low computational cost. We meet all these requirements by considering the difference between the k-th and $(k-1)$-th farthest distance of a $(k+1)$ nearest neighborhood, normalized by the $(k+1)$-th and k-th one respectively. Indeed, referring both distances to the same set, their difference helps to reduce the bias introduced by the edge effect as shown in our experimental results. Moreover, thanks to the different normalizations, the correlation between the two distances affects only marginally their joint distribution, so that we may still assume them to be independent random variables.

To achieve our goal, we consider a manifold $\boldsymbol{\mathcal{M}} \equiv \Re^d$ embedded in a higher dimensional space $\Re^D$ through a locally isometric nonlinear smooth map $\phi : \boldsymbol{\mathcal{M}} \to \Re^D$, and a sample set $\boldsymbol{X}_N = \{\boldsymbol{x}_i\}_{i=1}^N = \{\phi(\boldsymbol{z}_i)\}_{i=1}^N \subset \Re^D$, where $\boldsymbol{z}_i$ are independent identically distributed points drawn from $\boldsymbol{\mathcal{M}}$ according to a non-uniform smooth `pdf` $f : \boldsymbol{\mathcal{M}} \to \Re^+$.

To estimate the `id` of $\boldsymbol{\mathcal{M}}$, for each point $\boldsymbol{x}_i \in \boldsymbol{X}_N$ we find the set of $k+1$ $(1 \leq k \leq N-2)$ nearest neighbors $\bar{\boldsymbol{X}}_{k+1} = \bar{\boldsymbol{X}}_{k+1}(\boldsymbol{x}_i) = \{\boldsymbol{x}_j\}_{j=1}^{k+1} \subset \boldsymbol{X}_N$. Let r_{ij} denote the distance between $\boldsymbol{x}_j$ and $\boldsymbol{x}_i$, and, according to standard order statistics notation, refer to $r_{i(j)}$ as the j-th ordered smallest distance. The two normalized distances

$$\hat{r}_i = \frac{r_{i(k)}}{r_{i(k+1)}} \qquad \hat{r}'_i = \frac{r_{i(k-1)}}{r_{i(k)}} \tag{10}$$

are used to compute two vectors of normalized distances $\hat{\boldsymbol{r}} = \{\hat{r}_i\}_{i=1}^N$ and $\hat{\boldsymbol{r}}' = \{\hat{r}'_i\}_{i=1}^N$, together with their difference $\hat{\boldsymbol{\delta}} = \hat{\boldsymbol{r}} - \hat{\boldsymbol{r}}'$.

Observing that $\hat{\boldsymbol{\delta}}$ is a sample of an asymmetric Laplace distribution with parameters $\gamma = \sqrt{\frac{k-1}{k}}$ and $\sigma = \frac{1}{d}\sqrt{\frac{2}{k(k-1)}}$, we compute the statistics α and β according to Eq. (7). Equation (8) allows to obtain a `ML` estimate for $\hat{\sigma}$ which is used to get a twofold estimation for d as described in Eq. (9):

$$\hat{d}_1 = \frac{\sqrt{2}}{k\gamma\hat{\sigma}} \qquad \hat{d}_2 = \frac{\sqrt{2}\gamma}{(k-1)\hat{\sigma}} \tag{11}$$

Finally, we obtain a unique `id` estimate by computing the mean between $\hat{d}_1$ and $\hat{d}_2$ (i.e. $\hat{d} = (\hat{d}_1 + \hat{d}_2)/2$). We call this `id` estimator `DROS` (Dimensionality from Rank Order Statistics). Its time complexity is $O(DN \log N)$ and it is dominated by the time complexity of the `kNN` algorithm $(O(DN \log N))$. For the sake of clarity, in Appendix A the pseudocode of this algorithm is reported.

Considering Theorem 4 in [9], which ensures that geodetic distances in the infinitesimal ball converge to Euclidean distances with probability 1, Table 1 in [20], which resumes the convergence properties of `ML` estimators for asymmetric Laplace distributions, and the convergence results presented in Theorem 1, `DROS` represents a consistent estimator for the `id` of the manifold $\boldsymbol{\mathcal{M}}$.

5 Algorithm Evaluation

This section is organized as follows: in Sect. 5.1 we describe the datasets employed in our experiments; in Sect. 5.2 we summarize the adopted experimental settings; in Sect. 5.3 we report the results achieved by the proposed algorithm, comparing them to those obtained by state-of-the-art `id` estimators.

5.1 Dataset Description

To evaluate our algorithm, we performed experiments on the 10 synthetic and 4 real datasets reported in Table 1. In details, the 10 synthetic datasets were generated by employing the tool proposed in [15], while the used real datasets are the `ISOMAP` face database [29], the `MNIST` database [21], the `Santa Fe` [26] dataset, and the `DSVC1` time series [2].

Table 1. Short description of the 10 synthetic and 4 real datasets employed in our experiments, where d is the `id` and D is the embedding space dimension.

Dataset	Name	$\boldsymbol{d}$	$\boldsymbol{D}$	Description
Syntethic	$\mathcal{M}_1$	10	11	Uniformly sampled sphere linearly embedded
	$\mathcal{M}_2$	3	5	Affine space
	$\mathcal{M}_3$	4	6	Concentrated figure, confusable with a 3d one
	$\mathcal{M}_4$	4	8	Nonlinear manifold
	$\mathcal{M}_5$	2	3	2-d Helix
	$\mathcal{M}_6$	6	36	Nonlinear manifold
	$\mathcal{M}_7$	2	3	Swiss-Roll
	$\mathcal{M}_8$	1	13	Curve
	$\mathcal{M}_9$	10	11	Uniformly sampled hypercube
	$\mathcal{M}_{10}$	2	3	Möebius band 10-times twisted
Real	$\mathcal{M}_{\texttt{Faces}}$	3	4096	`ISOMAP` face dataset
	$\mathcal{M}_{\texttt{MNIST1}}$	$8-11$	784	`MNIST` database (digit 1)
	$\mathcal{M}_{\texttt{SantaFe}}$	9	50	`Santa Fe` dataset (version $D2$)
	$\mathcal{M}_{\texttt{DSVC1}}$	2.26	20	Real time series of a Chua's circuit

The `ISOMAP` face database consists in 698 gray-level images of size 64×64 representing the face of a sculpture. This dataset has three degrees of freedom, two for the pose and one for the lighting direction.

The `MNIST` database consists in 70000 gray-level images of size 28×28 of hand-written digits; in our tests we used the subset representing the digit 1, containing 6742 training points. The `id` of this database is not actually known, but we rely on the estimations proposed in [9,15] for the different digits, and in particular on the range $\{8..11\}$ for the digit 1.

The version $D2$ of the `Santa Fe` dataset is a synthetic time series of 50000 one-dimensional points; it was generated by a simulation of particle motion, and it has nine degrees of freedom. In order to estimate the attractor dimension of this time series, we used the method of delays described in [25], which generates D-dimensional vectors by collecting D values from the original dataset; by choosing $D = 50$ we obtained a dataset containing 1000 points in $\Re^{50}$.

The `DSVC1` is a real data time series composed of 5000 samples and measured from a hardware realization of the Chua's circuit [6]. We employed the method of delays choosing $D = 20$, and we obtained a dataset containing 250 points in $\Re^{20}$; the `id` of this dataset is ~ 2.26 as reported in [2].

5.2 Experimental Setting

To evaluate the quality of our method, we compared it with state-of-the-art `id` estimators such as: `BPCA`, `SPPCA`, `kNNG`, `CD`, and `MLE`. For `kNNG`, `MLE`, and `BPCA` we used the authors' implementation[4], while for the other algorithms we employed the version provided by the **Dimensionality Reduction Toolbox**[5] [31].

Table 2. Parameter settings for the different estimators: k represents the number of neighbors, γ is the edge weighting factor for `kNNG`, M is the number of Least Square (`LS`) runs, N is the number of resampling trials per `LS` iteration, α and π represent the parameters (shape and rate) of the Gamma prior distributions describing the hyper-parameters and the observation noise model of `BPCA`, μ contains the mean and the precision of the Gaussian prior distribution describing the bias inserted in the inference of `BPCA`.

Dataset	Method	Parameters
Synthetic	SPPCA	$None$
	BPCA	$iters = 500,\ \alpha = (2.0, 2.0)\ \pi = (2.0, 2.0)\ \mu = (0.0, 0.01)$
	CD	$None$
	MLE	$k_1 = 6\ k_2 = 20$
	kNNG	$k_1 = 6, k_2 = 20, \gamma = 1, M = 10, N = 1$
	DROS	$k = 12$
Real	SPPCA	$None$
	BPCA	$iters = 2000,\ \alpha = (2.0, 2.0)\ \pi = (2.0, 2.0)\ \mu = (0.0, 0.01)$
	CD	$None$
	MLE	$k_1 = 3\ k_2 = 8$
	kNNG	$k_1 = 3, k_2 = 8, \gamma = 1, M = 10, N = 1$
	DROS	$k = 10$

[4] http://www.eecs.umich.edu/~hero/IntrinsicDim/,
http://www.stat.lsa.umich.edu/~elevina/mledim.m,
http://research.microsoft.com/en-us/um/cambridge/projects/infernet/blogs/bayesianpca.aspx.

[5] http://cseweb.ucsd.edu/~lvdmaaten/dr/download.php.

As mentioned above, to create the synthetic datasets we adopted the generator described in [15] producing 20 instances of each dataset reported in Table 1, each of which is composed by 2500 randomly sampled points.

To obtain an unbiased estimation, for each technique we averaged the results achieved on the 20 instances. Furthermore, to execute multiple tests also on $\mathcal{M}_{\texttt{MNIST1}}$ and $\mathcal{M}_{\texttt{Isolet}}$ we extracted 5 random subsets containing 2500 points each, and we averaged the achieved results.

In Table 2 the configuration parameters employed in our tests are summarized. To relax the dependency of the `kNNG` algorithm from the selection of the value of its parameter k, we performed multiple runs with $k_1 \leq k \leq k_2$ and we averaged the obtained results (see Table 2).

5.3 Experimental Results

This section reports the results achieved on both synthetic and real datasets. In particular, Table 3 summarizes the results obtained on the synthetic datasets. It is possible to note that our technique strongly outperforms the projection methods we used as a baseline comparison; while it achieves results always comparable with those obtained by the best performing geometric approaches. In the last row of Table 3 the Mean Percentage Error (`MPE`) indicator, proposed in [24] in order to evaluate the overall performance of a given estimator, is reported. For each algorithm this value is computed as the mean of the percentage errors obtained on each dataset, i.e. $\texttt{MPE} = \frac{100}{\#\mathcal{M}} \sum_{\mathcal{M}} \frac{|\hat{d}_{\mathcal{M}} - d_{\mathcal{M}}|}{d_{\mathcal{M}}}$, where $d_{\mathcal{M}}$ is the real `id`, $\hat{d}_{\mathcal{M}}$ is the estimated one, and $\#\mathcal{M}$ is the number of tested manifolds. Considering this indicator, our method ranks as the best performing estimator.

Table 3. Results achieved on the synthetic datasets. The best approximations are highlighted in boldface.

Dataset	d	SPPCA	BPCA	kNNG	CD	MLE	DROS
$\mathcal{M}_8$	*1*	3.00	5.70	1.07	1.14	**1.00**	1.08
$\mathcal{M}_5$	*2*	3.00	**2.00**	2.06	1.98	1.97	2.02
$\mathcal{M}_7$	*2*	3.00	**2.00**	2.09	1.93	1.96	2.07
$\mathcal{M}_{10}$	*2*	3.00	1.55	**2.03**	2.19	2.21	2.05
$\mathcal{M}_2$	*3*	**3.00**	**3.00**	3.03	2.88	2.88	2.91
$\mathcal{M}_3$	*4*	**4.00**	**4.00**	3.82	3.23	3.83	3.89
$\mathcal{M}_4$	*4*	8.00	4.25	4.76	3.88	**3.95**	**4.05**
$\mathcal{M}_6$	*6*	12.00	12.00	11.24	**5.91**	6.39	6.27
$\mathcal{M}_1$	*10*	11.00	5.45	**9.89**	9.12	9.10	9.09
$\mathcal{M}_9$	*10*	**10.00**	5.20	10.21	8.09	8.26	9.21
MPE		56.00	69.22	13.10	8.36	5.64	**4.35**

In Table 4 the results obtained on real datasets are summarized[6]. As can be noticed, the MPE confirms that our technique is the best estimator. Moreover, these results show that our methodology is the most robust to the presence of noise, which usually characterizes real data.

Table 4. Results achieved on the real datasets by the employed approaches. The best approximations are highlighted in boldface.

Dataset	d	SPPCA	BPCA	kNNG	CD	MLE	DROS
$\mathcal{M}_{\texttt{DSVC1}}$	*2.26*	4.00	6.00	1.86	1.92	2.03	**2.23**
$\mathcal{M}_{\texttt{Faces}}$	*3*	5.00	4.00	4.32	**3.37**	4.05	4.03
$\mathcal{M}_{\texttt{Santa Fe}}$	*9*	19.00	18.00	7.43	4.39	7.16	**7.85**
$\mathcal{M}_{\texttt{MNIST1}}$	*8–11*	9.00	11.00	**9.58**	6.96	10.29	8.69
MPE		69.30	85.09	18.00	24.27	16.30	**15.97**

6 Conclusions and Future Works

In this paper we propose a novel consistent estimator, called DROS, that exploits neighboring distances to estimate the id of a given dataset by means of an efficient technique based on rank-order statistics.

Experiments on both synthetic and real datasets show that DROS is a really promising technique for id estimation, at least in terms of the Mean Percentage Error indicator.

Future works will be devoted both to identify a bound for the finite sample error to further formally evaluate the effectiveness of the proposed approach, and to effectively exploit on different scales the farthest neighbor distances to correctly deal with data drawn from manifolds of different ids. Moreover, it will be interesting to validate the quality of the proposed method by applying it as a parameter-estimation step of dimensionality reduction algorithms, and comparing the achieved results with those obtained by employing other state-of-the-art id estimators.

A Algorithm Implementation

In this appendix the pseudocode of our algorithm is reported. In Algorithm 1 DROS is shown, where $kNN(\boldsymbol{X}_N, \boldsymbol{x}, k)$ is the procedure that employs a k-nearest neighbor search returning the set of the k ordered nearest neighbors of $\boldsymbol{x}$ in $\boldsymbol{X}_N$ and their corresponding distances.

[6] Note that, when the true value of the id is not known, we considered the mean value of the range as $d_{\mathcal{M}}$.

Algorithm 1. Pseudo-code for the DROS algorithm.

```
Input:
    $\boldsymbol{X}_N$: The dataset points $\{\boldsymbol{x}_i\}_{i=1}^N$.
    $k$:        The kNN parameter.
Output:
    $\hat{d}$:          The estimated intrinsic dimensionality.

    {Define $\alpha$ and $\beta$.}
    $\alpha = 0$;
    $\beta = 0$;

    for i:=1 to N do begin
        {Find the ordered $k+1$ neighbors and their distances
        ($\boldsymbol{r}_i$) from $\boldsymbol{x}_i$.}
        $[\bar{\boldsymbol{X}}_{k+1}, \boldsymbol{r}_i] = kNN(\boldsymbol{X}_N, \boldsymbol{x}_i, k)$;

        {Normalize the $k$-order statistic by employing the $(k+1)^{th}$.}
        $\hat{\boldsymbol{r}}_i = \frac{\boldsymbol{r}_{i(k)}}{\boldsymbol{r}_{i(k+1)}}$;

        {Normalize the $(k-1)$-order statistic by employing the $k^{th}$.}
        $\hat{\boldsymbol{r}}'_i = \frac{r_{i(k-1)}}{r_{i(k)}}$;

        {Compute their difference.}
        $\hat{\boldsymbol{\delta}}_i = \hat{\boldsymbol{r}}_i - \hat{\boldsymbol{r}}'_i$;

        {Update $\alpha$.}
        $\alpha = \alpha + \frac{1}{N}\max\{\hat{\boldsymbol{\delta}}_i, 0\}$;

        {Update $\beta$.}
        $\beta = \beta - \frac{1}{N}\min\{\hat{\boldsymbol{\delta}}_i, 0\}$;
    end

    {Compute $\gamma$ and $\hat{\sigma}$.}
    $\gamma = \sqrt{\frac{k-1}{k}}$;
    $\hat{\sigma} = \sqrt{2}\,\sqrt[4]{\alpha}\,\sqrt[4]{\beta}\,(\sqrt{\alpha} + \sqrt{\beta})$;

    {Compute the id $\hat{d}$.}
    $\hat{d}_1 = \frac{\sqrt{2}}{k\gamma\hat{\sigma}}$;
    $\hat{d}_2 = \frac{\sqrt{2}\gamma}{(k-1)\hat{\sigma}}$;
    $\hat{d} = (\hat{d}_1 + \hat{d}_2)/2$;
```

References

1. Bishop, C.M.: Bayesian PCA. In: Proceedings of NIPS 11, pp. 382–388 (1998)
2. Camastra, F., Filippone, M.: A comparative evaluation of nonlinear dynamics methods for time series prediction. Neural Comput. Appl. **18**(8), 1021–1029 (2009)
3. Camastra, F., Vinciarelli, A.: Estimating the intrinsic dimension of data with a fractal-based method. IEEE Trans. PAMI **24**, 1404–1407 (2002)
4. Carter, K.M., Hero, A.O., Raich, R.: De-biasing for intrinsic dimension estimation. In: IEEE/SP 14th Workshop on Statistical Signal Processing, SSP 2007, pp. 601–605, Aug 2007
5. Ceruti, C., Bassis, S., Rozza, A., Lombardi, G., Casiraghi, E., Campadelli, P.: Danco: an intrinsic dimensionalty estimator exploiting angle and norm concentration. Elsevier, Pattern Recogn. **47**(8), 2569–2581 (2014)

6. Chua, L., Komuro, M., Matsumoto, T.: The double scroll. IEEE Trans. Circuits Syst. **32**, 797–818 (1985)
7. Costa, J.A., Hero, A.O.: Geodesic entropic graphs for dimension and entropy estimation in manifold learning. IEEE Trans. Sig. Process. **52**(8), 2210–2221 (2004)
8. Costa, J.A., Hero, A.O.: Learning intrinsic dimension and entropy of high-dimensional shape spaces. In: Proceedings of EUSIPCO (2004)
9. Costa, J.A., Hero, A.O.: Learning intrinsic dimension and entropy of shapes. In: Statistics and Analysis of Shapes, Birkhauser (2005)
10. Friedman, J.H., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning - Data Mining, Inference and Prediction. Springer, Berlin (2009)
11. Fukunaga, K.: Intrinsic dimensionality extraction. In: Krishnaiah, P.R., Kanal, L.N. (eds.) Classification, Pattern Recognition and Reduction of Dimensionality. North Holland, Amsterdam (1982)
12. Fukunaga, K., Olsen, D.R.: An algorithm for finding intrinsic dimensionality of data. IEEE Trans. Comput. **20**, 176–183 (1971)
13. Grassberger, P., Procaccia, I.: Measuring the strangeness of strange attractors. Physica D: Nonlinear Phenom. **9**, 189–208 (1983)
14. Guan, Y., Dy, J.G.: Sparse probabilistic principal component analysis. J. Mach. Learn. Res. - Proc. Track **5**, 185–192 (2009)
15. Hein, M.: Intrinsic dimensionality estimation of submanifolds in euclidean space. In: Proceedings of ICML, pp. 289–296 (2005)
16. Johnson, N.L., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions, vol. 2. Wiley, New York (1995)
17. Jollife, I.T.: Adaptive Control Processes: A Guided Tour. Princeton University Press, Princeton (1961)
18. Jollife, I.T.: Principal Component Analysis. Springer Series in Statistics. Springer, New York (1986)
19. Kirby, M.: Geometric Data Analysis: an Empirical Approach to Dimensionality Reduction and the Study of Patterns. Wiley, New York (1998)
20. Kotz, S., Kozubowski, T.J., Podgórski, K.: Maximum likelihood estimation of asymmetric laplace parameters. Ann. Inst. Stat. Math. **54**(4), 816–826 (2002)
21. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**, 2278–2324 (1998)
22. Levina, E., Bickel, P.J.: Maximum likelihood estimation of intrinsic dimension. In: Proceedings of NIPS 17(1), pp. 777–784 (2005)
23. Li, J., Tao, D.: Simple exponential family PCA. In: Proceedings of AISTATS, pp. 453–460 (2010)
24. Lombardi, G., Rozza, A., Ceruti, C., Casiraghi, E., Campadelli, P.: Minimum neighbor distance estimators of intrinsic dimension. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 374–389. Springer, Heidelberg (2011)
25. Ott, E.: Chaos in Dynamical Systems. Cambridge University Press, Cambridge (1993)
26. Pineda, F.J., Sommerer, J.C.: Estimating generalized dimensions and choosing time delays: A fast algorithm. In: Time Series Prediction. Forecasting the Future and Understanding the Past, pp. 367–385 (1994)
27. Rozza, A., Lombardi, G., Ceruti, C., Casiraghi, E., Campadelli, P.: Novel high intrinsic dimensionality estimators. Mach. Learn. J. **89**(1–2), 37–65 (2012)
28. Rozza, A., Lombardi, G., Rosa, M., Casiraghi, E., Campadelli, P.: IDEA: intrinsic dimension estimation algorithm. In: Maino, G., Foresti, G.L. (eds.) ICIAP 2011, Part I. LNCS, vol. 6978, pp. 433–442. Springer, Heidelberg (2011)

29. Tenenbaum, J., Silva, V., Langford, J.: A global geometric framework for nonlinear dimensionality reduction. Science **290**, 2319–2323 (2000)
30. Tipping, M.E., Bishop, C.M.: Probabilistic principal component analysis. J. Royal Stat. Soc., Ser. B **61**(Pt. 3), 611–622 (1997)
31. Van der Maaten, L.J.P.: An introduction to dimensionality reduction using matlab. Technical report, Delft University of Technology (2007)
32. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
33. Verveer, P.J., Duin, R.P.W.: An evaluation of intrinsic dimensionality estimators. IEEE Trans. PAMI **17**, 81–86 (1995)
34. Wilks, S.S.: Mathematical Statistics. Wiley Publications in Statistics. John Wiley, New York (1962)

Dimensionality Reduction in Boolean Data: Comparison of Four BMF Methods

Eduard Bartl[1], Radim Belohlavek[1], Petr Osicka[1](✉), and Hana Řezanková[2]

[1] Data Analysis and Modeling Laboratory (DAMOL), Department of Computer Science, Palacky Univeristy, Olomouc, Czech Republic
eduard.bartl@upol.cz, {radim.belohlavek,osicka}@acm.org

[2] Department of Statistics and Probability, Faculty of Informatics and Statistics, University of Economics, Prague, Czech Republic
rezanka@vse.cz

Abstract. We compare four methods for Boolean matrix factorization (BMF). The oldest of these methods is the 8M method implemented in the BMDP statistical software package developed in the 1960s. The three other methods were developed recently. All the methods compute from an input object-attribute matrix I two matrices, namely an object-factor matrix A and a factor-attribute matrix B in such a way that the Boolean matrix product of A and B is approximately equal to I. Such decompositions are utilized directly in Boolean factor analysis or indirectly as a dimensionality reduction method for Boolean data in machine learning. While some comparison of the BMF methods with matrix decomposition methods designed for real valued data exists in the literature, a mutual comparison of the various BMF methods is a severely neglected topic. In this paper, we compare the four methods on real datasets. In particular, we observe the reconstruction ability of the first few computed factors as well as the number of computed factors necessary to fully reconstruct the input matrix, i.e. the approximation to the Boolean rank of I computed by the methods. In addition, we present some general remarks on all the methods being compared.

1 Matrix Decompositions, Dimensionality Reduction, and Boolean Data

Matrix Decompositions. Various matrix decomposition methods have nowadays an established role in science and engineering. Probably the most widely used and studied are the singular value decomposition, principal component analysis, independent component analysis, and various factor analysis methods [6,11, 12,17], which were designed for real-valued matrices. Let us also mention the

E. Bartl and H. Rezankova acknowledge support by Grant No. 202/10/0262 of the Czech Science Foundation. R. Belohlavek and P. Osicka acknowledge support by the ESF project No. CZ.1.07/2.3.00/20.0059, the project is co-financed by the European Social Fund and the state budget of the Czech Republic.

F. Masulli et al. (Eds.): CHDD 2012, LNCS 7627, pp. 118–133, 2015.
DOI: 10.1007/978-3-662-48577-4_8

non-negative matrix factorization [5,11,14] which aims at decomposing a non-negative matrix into a product of two non-negative matrices. The latter constrain makes the method related to the Boolean matrix factorization, yet, the calculus behind this method is the ordinary matrix calculus.

Boolean Data. Long before the interest in Boolean data in data mining, it has been recognized that the nature of Boolean data (also called binary data, presence-absence data, 0/1 data or the like) requires a specific treatment, see e.g. [23,24]. The literature on the analysis of Boolean data using matrix methods reveals, briefly speaking, the following picture. First, methods designed for real-valued data, such as the ones mentioned above, are applied to Boolean data, see e.g. [31] for more information. Second, methods designed originally for real-valued data are modified for Boolean data, see e.g. [15,26–28,30,33]. As pointed out in several papers, see e.g. [20,31], the methods which bring a Boolean matrix into a product of real-valued matrices with possibly negative values, such as those mentioned above, suffer the problem of interpretability of the output matrix entries. Unlike methods mentioned above, the methods examined in the present paper are based on the calculus of Boolean matrices [13]. In particular, the methods aim at finding for a given Boolean matrix I two Boolean matrices, A and B, whose Boolean product is equal or approximately equal to I. Such methods have been investigated in several papers, see e.g. [4,8,18–20]. Note also that computational complexity aspects of the problems related to the decomposition of Boolean matrices was investigated for the first time in [29] (in different terms, however) and is further discussed e.g. in [20,21,32].

Dimensionality Reduction. A common feature of matrix decompositions (factorizations) in general and those designed for Boolean data in particular is dimensionality reduction. Namely, a decomposition of an input $n \times m$ object-variable matrix I into a product of an $n \times k$ and $k \times m$ matrices A and B corresponds to a discovery of k new variables, called factors and interpret A and B as the object-factor and factor-variable matrices. In such case, the original relationship between objects and variables described by I may be reconstructed from the relationship between objects and factors described by A and the relationship between factors and the original variables described by B. Thus, if $k < m$, it is reasonable to consider the factors as more fundamental variables, which are "hidden" in that input data. The objects may be represented in the lower-dimensional (k-dimensional) space of factors instead of the m dimensional space of the original variables and using B, one may transform an object representation in the factor space to the variable space and vice versa. This general logic applies to Boolean data and Boolean matrix factorizations (BMF) as well. The dimensionality reduction has two important aspects. First, a technical one which consists in the possibility to represent and process objects in a lower-dimensional space, resulting in faster processing and less storage space. Closely connected is the second aspect, namely the knowledge discovery aspect. The factors very often represent more fundamental variables of which the original variables are just particular manifestations. In this regard, one benefit consists in the fact that the discovery of factors itself provides a useful information for the user: the user gets to know the “true” explanatory variables for the input data. Another benefit is that as a rule, processing the data in terms of factors leads to improvement in quality compared to processing in terms of the original variables which are in fact derived from the factors. The reader interested in particular applications of the above-mentioned benefits of dimensionality reduction due to BMF is referred to [4,10,16,20,24,25,31,32].

In the present paper, we compare four selected methods for decomposition of Boolean matrices. In Sect. 2, we describe the problem of Boolean matrix factorization and its variants. In Sect. 3, we briefly describe the four methods compared in this paper, including the 8M which seems to be unknown to the recent literature and seems not to be described in the literature The experimental comparison of the methods is presented in Sect. 4. Section 5 contains a discussion on further issues and concludes the paper.

2 Boolean Matrix Factorization

General BMF Problem. We denote by I an $n \times m$ Boolean matrix, i.e. the entry I_{ij} corresponding to row i and column j is either 0 or 1. We use I because very often, the matrix describes an *incidence* relation between n objects, which correspond to the rows, and m attributes, which correspond to the columns of I. The set of all $n \times m$ Boolean matrices is denoted by $\{0,1\}^{n\times m}$. The ith column and jth row of I are denoted by $I_{i\_}$ and $I_{\_j}$, respectively. A general aim in BMF is to find for a given $I \in \{0,1\}^{n\times m}$ (and possibly other parameters) matrices $A \in \{0,1\}^{n\times k}$ and $B \in \{0,1\}^{k\times m}$ for which

$$I \text{ is (approximately) equal to } A \circ B, \tag{1}$$

where $\circ$ denote the Boolean matrix product given by

$$(A \circ B)_{ij} = \max_{l=1}^{k} \min(A_{il}, B_{lj}). \tag{2}$$

Such an exact or approximate decomposition of I into $A \circ B$ corresponds to a discovery of k factors (new Boolean variables) that exactly or approximately explain the data: $A_{il} = 1$ indicates that factor l applies to object i while B_{lj} indicates that attribute j is a particular manifestation of factor l (think of person P as object, “being fluent in English” as attribute, and “having good education” as factor). The least k for which an exact decomposition $I = A \circ B$ exists is called the Boolean (or Schein) rank of I [4,13,20]. Then, according to (2), the factor model (1) reads:

> object i has attribute j if and only if there exists factor l such that l applies to i and j is a particular manifestation of l.

The matrices I, A, and B are usually called the object-attribute matrix, the object-factor (or factor score) matrix, and the factor-attribute (or factor loadings or basis vector) matrix [4,20].

Matrix Distance and Error Measurement. To assess how well the product $A \circ B$ approximates the input matrix I, we need an appropriate matrix distance function. A natural choice is the following. Recall that the L_1-norm (Hamming weight in case of Boolean matrices) of an $n \times m$ matrix C is given by

$$||C|| = \sum_{i=1,j=1}^{m,n} |C_{ij}|.$$

The induced matrix distance, that is conveniently used for measuring error in matrix decompositions [13, 20], is given by

$$E(C, D) = ||C - D|| = \sum_{i=1,j=1}^{m,n} |C_{ij} - D_{ij}|. \quad (3)$$

Two Particular Problems. The following particular variants of the general BMF problem have been discussed in the literature.

Problem 1.
input: Boolean matrix $I \in \{0,1\}^{n\times m}$, positive integer k
output: Boolean matrices $A \in \{0,1\}^{n\times k}$ and $B \in \{0,1\}^{k\times m}$ minimizing $||I - A \circ B||$.

This problem is called the *discrete basis problem* (DBP) in [20]. In [4], the following the problem is considered:

Problem 2.
input: Boolean matrix $I \in \{0,1\}^{n\times m}$, positive integer ε
output: Boolean matrices $A \in \{0,1\}^{n\times k}$ and $B \in \{0,1\}^{k\times m}$ with k as small as possible such that $||I - A \circ B|| \leq \varepsilon$.

The two problems reflect two important views on BMF, the first one emphasizing the importance of the first k (presumably most important) factors, the second one emphasizing the need to account for (and thus to explain) a prescribed portion of data. We use these two problems as the main ones for comparing the two methods in our paper.

An important fact is that both Problem 1 and Problem 2 are NP-hard optimization problems (see e.g. [4, 20, 32]) and that the hardness of this problem follows from NP-hardness of the set basis problem [29]. As a result, unless P = NP, efficient algorithms attempting to solve Problem 1 and 2 need to designed as approximation algorithms.

3 The Four Methods Being Compared

In this section, we briefly describe the four methods compared in this paper.

Algorithm 1. This algorithm, described in [4], utilizes formal concepts of I as factors. Recall that a formal concept of I (see [9]) is any pair $\langle C, D\rangle$ of sets $C \subseteq \{1,\ldots,n\}$ (rows, objects) and $D \subseteq \{1,\ldots,n\}$ (columns, attributes) satisfying the following property: D is the set of all attributes j for which $I_{ij} = 1$ for every object $i \in C$ and, vice versa, C is the set of all objects i for which $I_{ij} = 1$ for every attribute $j \in D$. Formal concepts are very well understood by domain experts.

Note that formal concepts may be thought of as particular biclusters in I and that a closer examination of the relationships between the problems involved in BMF and those involved in biclustering might be worth pursuing. Geometrically, they are, up to permuting rows and columns, just maximal rectangles full of 1s in the matrix I. If a set $\mathcal{F}$ of formal concepts is to be used as a set of factors of I, the corresponding matrices $A_{\mathcal{F}}$ and $B_{\mathcal{F}}$ are defined the following way: column l of $A_{\mathcal{F}}$ is just the characteristic vector of C_l and row l of $B_{\mathcal{F}}$ is just the characteristic vector of D_l, where $\langle C_l, D_l\rangle$ is the lth formal concept in $\mathcal{F}$. It is proved in [4] that using such factors is optimal in that the Boolean rank of I may be achieved by using formal concepts as factors. In Algorithm 1, one first computes all the formal concepts of I. The algorithm proceeds in a greedy way: In every step, it selects the concept that covers the largest number of entries with 1 in I that were not covered by the previously selected concepts ($\langle C, D\rangle$ covers $I_{ij} = 1$ if $i \in C$ and $j \in D$, i.e. the rectangle corresponding to $\langle C, D\rangle$ spans over the entry $\langle i, j\rangle$).

Algorithm 2. This algorithm, described in [4], utilizes formal concepts of I as factors the same way as Algorithm 1. However, Algorithm 2 avoids the necessity to compute all the concepts of I and browse through them during the greedy selection. Instead, the algorithm computes the candidate factors, i.e. concepts of I, on demand the following, greedy way. Each time a new factor is needed, one looks at the columns of I and selects the one concept generated by a column which covers most of the yet uncovered 1s in I. Such a concept corresponds to a narrow but high rectangle in the data. Then one tries to see if such rectangle may be extended to a wider (and thus not so high) rectangle by adding some attribute and deleting the objects so that one still has a rectangle. If so, one selects the best such rectangle, i.e. covering most of the yet uncovered 1s in I. One repeats the process of extension until no such extension yields a better rectangle. This way one obtains the new factor and eventually a set $\mathcal{F}$ of formal concepts—the factors of I. For both, Algorithm 1 and 2, the resulting set $\mathcal{F}$ of concepts has always the property $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$, i.e. the algorithms find an exact decomposition of I. Hence, Algorithm 1 and 2 can be used for Problem 1 and 2. Namely, one stops the algorithm after k factors are computed (Problem 1) or after the error drops below ε, i.e. after $||I - A_{\mathcal{F}} \circ B_{\mathcal{F}}|| \leq \varepsilon$ (Problem 2).

Asso. The Asso algorithm [20] works as follows. From the input $n \times m$ matrix I, the required number k of factors, and additional parameters τ, w^+, and w^-, the algorithm computes an $m \times m$ Boolean matrix C in which $C_{ij} = 1$ if the confidence of the association rule $\{i\} \Rightarrow \{j\}$ is at least τ, i.e. if $a/b \geq \tau$, where a is the number of rows r of I in which $I_{ri} = I_{rj} = 1$ and b is the number of rows r of I in which $I_{ri} = 1$. The rows of C are then the candidate rows for matrix B. The actual k rows of B are selected from the rows of C in a greedy manner using parameters w^+ and w^-. For every candidate row $C_{l\_}$ of C (candidate for a new row of B), one computes the best corresponding column c (candidate for a new column of A), i.e. a c maximizing function $cover(C_{l\_}, c) = w^+ \cdot cov - w^+ \cdot err$ where cov is the number of entries $\langle i, j\rangle$ covered by $C_{l\_}$ and c (in that $c_i = 1$

and $C_{lj} = 1$) which were not covered by the previously selected factors (drop in error due to $C_{l\_}$ and c), and *err* is the number of entries $\langle i, j\rangle$ for which $I_{ij} = 0$ but are covered by $C_{l\_}$ and c, and not covered by any of the previously selected factors (increase in error due to $C_{l\_}$ and c). One then adds to A and B as a new column and row the c and $C_{l\_}$ for which $cover(C_{l\_}, c)$ is largest but positive. If k factors have been computed or if no l with positive $cover(C_{l\_}, c)$ exists, one stops. Asso is designed for Problem 1 but generally it cannot be used for Problem 2 because there is no guarantee in general that matrices A and B with $||I - A_{\mathcal{F}} \circ B_{\mathcal{F}}|| \leq \varepsilon$ are found. In particular, Asso may not be used for computing an exact decomposition of I.

8M of BMDP. This method is implemented in a quite old but still very good statistical package BMDP [18], developed at the UCLA in the 1960s, originally for biomedical applications. The 8M method (called also Boolean Factor Analysis method in BMDP) is very interesting and seems to be unknown to the recent literature on BMF. The algorithm behind 8M works as follows. To compute k factors of I (and thus the corresponding $n \times k$ and $k \times m$ matrices A and B), the algorithm starts $q < k$ candidate rows of B (candidate factor loadings). These are either supplied by the user or computed from I using a heuristic based on inclusion of the columns of I. From this set of q factor loading, the algorithm computes q factor scores (candidate columns of A); from the q scores, the algorithm tries to find better q factor loadings, etc. until no change occurs or three such cycles are completed. Such tuning of factor loadings and scores is called refinement (the details are too technical to be included here). The algorithm then iteratively adds further factors as follows. Suppose l factors have been obtained. Then, one adds new factor $l + 1$, refines the loadings and scores of all the factors as above, adds new factor $l + 2$ and refines again. Then the lth factor is removed and the remaining factors are refined. Consequently, the process is repeated, i.e. two new factors are added, one is removed, etc. For example, starting with $q = 2$ factors, we obtain 2, 3, 4, 3, 4, 5, 4, 5, 6, 7, 6, 7, 8, etc. factors. The process stops when the required number k of factors is obtained the second time. For example, with $q = 2$ and $k = 6$, one computes 2, 3, 4, 3, 4, 5, 4, 5, 6, 7, 6 factors and the last six ones are the final factors output by the algorithm. By default, $q = k - 2$ but q may be set by the user. The algorithm therefore performs steps back to possibly decrease E_o component of error. A new factor is added based on the matrix describing the error committed by the factors obtained so far. In particular, one uses the column of I which contains the largest number of 1s uncovered by the previously computed factors. As with Asso, the significance of the two types of errors (see Sect. 4.1) is controlled by a parameter (by default, both types have the same weight). It follows from the description that 8M is designed for Problem 1. Note that the actual scenario of the run of the 8M method in BMDP may be controlled by user-specified parameters, but default of these parameters values make it possible for the user to run the method by just supplying the input data.

4 Experimental Comparison

4.1 Method of Comparison

It follows from the above-mentioned facts that if A and B form the output of any BMF method, one may look at the factors as rectangles full of 1s. Such rectangles should approximately cover the input matrix I which is due to the objective to have small error $E(I, A \circ B)$, see (3). Clearly, E may be seen as being the sum of two components, E_u corresponding to 1s in I that are 0 s in $A \circ B$ (uncovered) and E_o corresponding to 0s in I that are 1s in $A \circ B$ (overcovered):

$$E(I, A \circ B) = E_u(I, A \circ B) + E_o(I, A \circ B) \tag{4}$$

with

$$E_u(I, A \circ B) = |\{\langle i, j\rangle \mid I_{ij} = 1, (A \circ B)_{ij} = 0\}|,$$
$$E_o(I, A \circ B) = |\{\langle i, j\rangle \mid I_{ij} = 0, (A \circ B)_{ij} = 1\}|.$$

Note that in the BMDP manual on the 8M method [18], E_u and E_o are called the positive and negative discrepancy, respectively. For convenience, we use the functions

$$e_u(I, A \circ B) = \frac{E_u(I, A \circ B)}{||I||} \text{ and } e_o(I, A \circ B)) = \frac{E_o(I, A \circ B)}{||I||}$$

and

$$e(I, A \circ B) = e_u(I, A \circ B) + e_o(I, A \circ B)) = \frac{E(I, A \circ B)}{||I||}$$

measuring a kind of a relative error of $A \circ B$ with respect to I, and

$$q(I, A \circ B) = 1 - e(I, A \circ B). \tag{5}$$

which may be thought of as measuring coverage quality. Clearly, $q(I, A \circ B) = 1$ if and only if $I = A \circ B$ (exact decomposition). Furthermore, $q(I, A \circ B)$ decreases with increasing error, i.e. with increasing $E(I, A \circ B)$.

4.2 Datasets Used

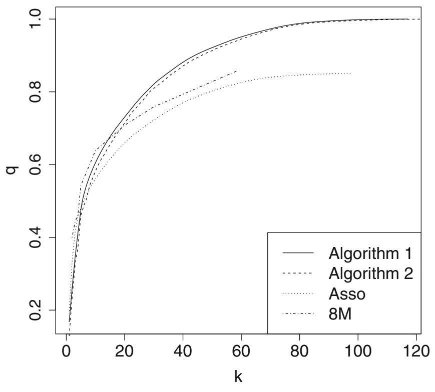
We used the datasets described in Table 1.[1] The columns Rows and Columns show the dimension of matrix I describing the dataset (indicating n rows and m columns), No. 1s contains the number of entries in the input matrix equal to 1 (i.e. $||I||$), and Density is the percentage of 1s (i.e. $\frac{||I||}{n \cdot m}$).

The *Mushroom* dataset is taken from the UCI Machine Learning Repository [1]. The original dataset consists of 8125 objects and 23 nominal attributes (for example, attribute *class* with values *edible* and *poisonous*, or attribute *cap-shape* taking values such as *bell*, *conical* or *convex*). We transformed this dataset to a

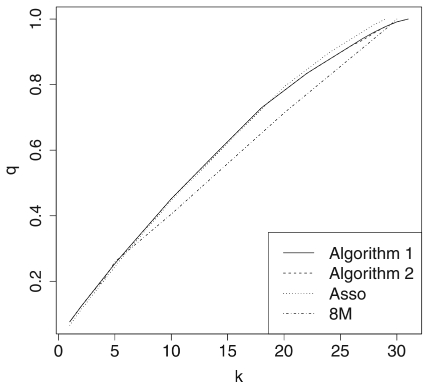
[1] We thank P. Miettinen for providing us with the datasets DBLP, DNA, and Paleo.

Table 1. Real datasets

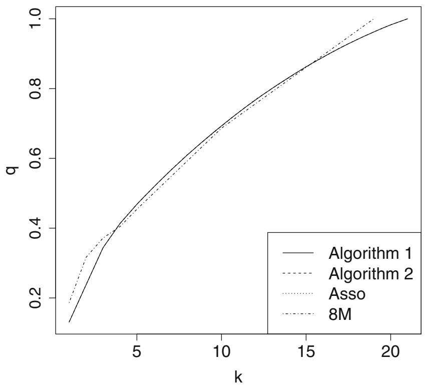
Dataset	Rows	Columns	no. 1s	Density
Mushroom	8124	119	186852	0.19
Tic-tac-toe	959	30	9580	0.33
DBLP	19	6980	17179	0.13
DNA	4590	393	26527	0.01
House Votes Republicans	108	16	907	0.52
House Votes Democrats	124	16	1032	0.52
Paleo	501	139	3537	0.05



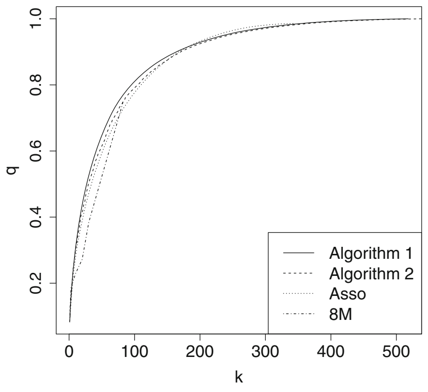
(a) Mushroom



(b) Tic-Tac-Toe

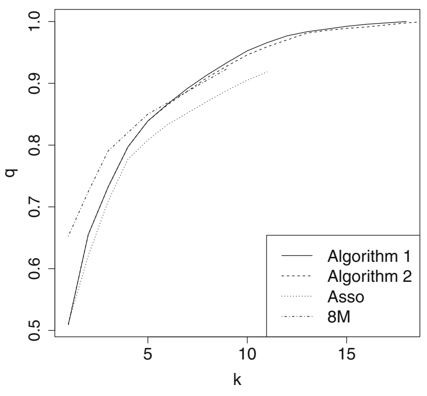


(c) DBLP

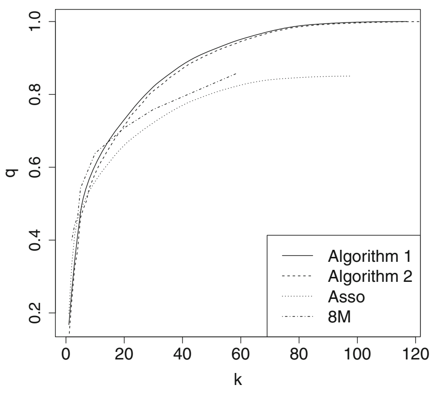


(d) DNA

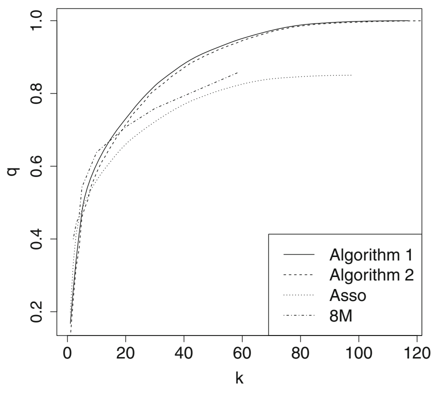
Fig. 1. Performance of algorithms on selected real datasets (relative error $q(I, A \circ B)$ where A and B are given by the first k factors)



(a) House Votes Republicans



(b) House Votes Democrats



(c) Paleo

Fig. 2. Performance of algorithms on selected real datasets (relative error $q(I, A \circ B)$ where A and B are given by the first k factors)

Boolean matrix by nominal scaling, i.e. by replacing a nominal attribute y with p values $v_1, \ldots, v_p$ by p Boolean attributes $y_{v_1}, \ldots, y_{v_p}$ in such a way that at ith row, the value of the column corresponding to y_{v_j} is 1 if and only if the value of the attribute y at ith row in the original dataset is equal to v_j.

The *Tic-tac-toe* dataset is taken from [1] as well, and it encodes the complete set of possible board configurations at the end of tic-tac-toe games. The original dataset consists of 958 objects and 9 nominal attributes (each attribute taking one of the three values: x, o, *blank*). Binarization of this dataset was performed by nominal scaling as described above.

The *DBLP* dataset contains information about in which of the 19 selected conferences the 6980 authors have published. The dataset is based on the DBLP database (http://www.informatik.uni-trier.de/~ley/db/).

Table 2. Performance of algorithms on real datasets I (coverage quality by first k factors)

Dataset	k	Algorithm 1	Algorithm 2	Asso	8M (BMDP)
Mushroom	1	0.1678	0.1294	0.1988	-
	2	0.2603	0.2335	0.3333	0.4002
	3	0.3435	0.3231	0.4138	0.4422
	4	0.4129	0.3724	0.4570	0.4630
	5	0.4750	0.4608	0.4807	0.5427
	10	0.6044	0.5823	0.5606	0.6383
	20	0.7307	0.7148	0.6602	0.7081
	30	0.8210	0.8096	0.7221	0.7588
	59	0.9415	0.9486	0.8235	**0.8591**
	98	0.9977	0.9955	**0.8502**	-
	117	**1.0000**	0.9994	-	-
	128	-	**1.0000**	-	-
Tic-tac-toe	1	0.0764	0.0764	0.0653	0.0764
	2	0.1229	0.1229	0.1131	0.1230
	3	0.1665	0.1665	0.1567	0.1666
	4	0.2102	0.2102	0.2004	0.2042
	5	0.2538	0.2538	0.2440	0.2595
	10	0.4517	0.4517	0.4455	0.4048
	20	0.7816	0.7816	0.7933	0.7135
	29	0.9771	0.9771	**1.0000**	0.9693
	30	0.9914	0.9914	-	**1.0000**
	31	**1.0000**	**1.0000**	-	-
DBLP	1	0.1312	0.1312	0.1312	0.1867
	2	0.2378	0.2378	0.2378	0.3165
	3	0.3439	0.3439	0.3439	0.3717
	4	0.4131	0.4131	0.4131	0.4044
	5	0.4683	0.4683	0.4683	0.4541
	10	0.6923	0.6923	0.6923	0.6865
	19	0.9624	0.9624	0.9624	**1.0000**
	20	0.9827	0.9827	0.9827	-
	21	**1.0000**	**1.0000**	**1.0000**	-
DNA	1	0.0827	0.0827	0.0880	0.0950
	2	0.1303	0.1303	0.1137	0.1479
	3	0.1615	0.1563	0.1405	0.1861
	4	0.1881	0.1875	0.1655	0.1955
	5	0.2141	0.2128	0.1893	0.1996
	10	0.3074	0.3008	0.2795	0.2297
	20	0.4383	0.4145	0.3890	0.2720
	30	0.5250	0.5031	0.4716	0.3835
	83	0.7706	0.7503	0.7288	**0.7523**
	345	0.9839	0.9817	**0.9858**	-
	518	**1.0000**	0.9992	-	-
	539	-	**1.0000**	-	-

Table 3. Performance of algorithms on real datasets II (coverage quality by first k factors)

Dataset	k	Algorithm 1	Algorithm 2	Asso	8M (BMDP)
House Votes Rep.	1	0.5093	0.5093	0.5115	0.6527
	2	0.6549	0.6549	0.6207	0.7244
	3	0.7320	0.7320	0.7089	0.7905
	4	0.7971	0.7971	0.7772	0.8203
	5	0.8390	0.8390	0.8081	0.8501
	9	0.9338	0.9283	0.8886	**0.9239**
	10	0.9525	0.9459	0.9051	-
	11	0.9658	0.9592	**0.9184**	-
	18	**1.0000**	0.9977	-	-
	19	-	**1.0000**	-	-
House Votes Dem.	1	0.3682	0.3682	0.2819	0.5155
	2	0.4641	0.4641	0.3905	0.6143
	3	0.5523	0.5523	0.4844	0.7006
	4	0.6240	0.6240	0.5562	0.7422
	5	0.6937	0.6937	0.6269	0.7694
	10	0.8953	0.8895	0.9001	**0.8886**
	14	0.9534	0.9486	**0.9806**	-
	20	0.9932	0.9874	-	-
	24	**1.0000**	0.9980	-	-
	26	-	**1.0000**	-	-
Paleo	1	0.0265	0.0265	0.0265	0.0266
	2	0.0486	0.0474	0.0474	0.0475
	3	0.0695	0.0695	0.0684	0.0746
	4	0.0890	0.0890	0.0879	0.0792
	5	0.1063	0.1063	0.1051	0.0871
	10	0.1815	0.1809	0.1815	0.0978
	20	0.3138	0.3101	0.3135	0.1866
	30	0.4212	0.4209	0.4240	0.2485
	135	0.9722	0.9663	0.9892	**0.9706**
	139	0.9802	0.9748	**0.9997**	-
	156	**1.0000**	0.9994	-	-
	157	-	**1.0000**	-	-

The *DNA* dataset is taken from [22] and contains binary chromosome subband-specific information of DNA amplifications in human neoplasms.

The *House Votes Republicans* and *House Votes Democrats* datasets are taken from the UCI Machine Learning Repository [1]. The original dataset contains 168

Republican and 267 Democratic voting results on the 16 issues (e.g. education spending, duty free exports or immigration). We divided the dataset into two, one for Republicans, one for Democrats. Then we removed from both of them all the rows with missing values, reducing the size to 108×16 and 124×16, respectively.

The *Paleo* dataset comes from [7] and describes fossil records per location. The data is based on NOW public release 030717.

4.3 Results

We ran Algorithms 1 and 2 to let them compute the exact decomposition of the input matrix I. From the resulting set of factors, one may then directly obtain the coverage quality (5) of the first l factors for $l = 1, \dots, k$, where k is the number of factors computed by the algorithm.

For Asso, we ran the algorithm for $k = m$, i.e. we asked Asso to compute at most m (number of attributes) factors. Due to the logic of Asso, from the computed k factors, we may again directly obtain the coverage quality (5) of the first l factors. This is because, as in case of Algorithm 1 and 2, when letting Asso compute k_1 factors and then, starting again, letting it compute $k_2 < k_1$ factors, the sequence of the first k_2 factors computed is the same for both runs. Asso requires to set parameters τ, w^+, and w^-. In fact, since w^+ and w^- are connected, we kept $w^+ = 1$ and varied w^- only. For every dataset, we ran Asso with all possible combinations of $\tau \in \{0.85, 0.9, 0.95\}$ and $w^- \in \{1, 2, 3\}$ and selected the parameters for which Asso performed best. We report the results based on the best performance of Asso obtained this way. The parameters are given by Table 4.

Table 4. Parameters for Asso algorithm

Dataset	τ	w^+	w^-
Mushroom	0.95	1	2
Tic-tac-toe	0.9	1	1
DBLP	0.9	1	2
DNA	0.95	1	3
House Votes Republicans	0.9	1	3
House Votes Democrats	0.95	1	1
Paleo	0.95	1	1

We ran the 8M method with the default parameters. Due to the logic of 8M, the method needs to be run again from start for every required number k of attributes because when computing new factors, it recomputes the factors computed so far.

This way, we are able to observe from the results how the methods perform in solving both Problem 1 (fix the number k of factors and observe the corresponding coverage quality q achieved by the respective algorithm; q corresponds uniquely to the error function E) and Problem 2 (fix coverage quality q, which corresponds to ε, and observe the corresponding number k of factors needed by the respective algorithm).

The results are depicted in the graphs in Figs. 1 and 2 and in Tables 2 and 3. Due to limited scope, we include only results regarding the coverage quality (5), which is nevertheless the most important measure. For some input data and values, the BMDP software failed to compute the results, basically due to the data size issues. Because of the properties of Asso, namely the fact, that Asso is generally not able to compute exact decompositions, some values are missing for Asso as well.

The graphs depict for each dataset involved the coverage quality (5) of each of the four algorithms (vertical axis) of the k factors computed by the algorithms (horizontal axis). The results further support the observation from [4] that even though Algorithm 2 is designed as a simplified version of Algorithm 1 in that it searches through a limited number of candidate factors, in terms of quality of the computed factors, the two algorithms have nearly the same performance. For 4 out of 7 datasets (*DBLP, DNA, Paleo, Tic-tac-toe*) also the performance of Asso is very similar to Algorithm 1 and Algorithm 2, although Asso shows a tendency to perform slightly worse than the other two methods for small numbers of factors, and on the other hand, to perform slightly better for higher numbers of factors. A clear example of this behaviour is the dataset *House Votes Democrats*. For the remaining two datasets (*Mushroom, House Votes Republicans*) Asso is the worst of all the compared methods. The results we have for 8M of BMDP indicate that this method computes for both *House Votes* datasets and for a small number of factors a decomposition with the highest coverage quality. However, as the number of factors grows its performance in comparison with the remaining methods declines.

Let us also note that even though we did not perform a careful comparison of time efficiency of the algorithms, basically because we did not use our own implementation of 8M, the following may be observed. The fastest algorithm is Algorithm 2 which implements a direct greedy selection without any preprocessing. Second fastest is Asso which needs to preprocess the data, followed by 8M, and Algorithm 2, which is slowed down very much by computing the possibly large set of all formal concepts of the input matrix first, and then iterating through it multiple times.

5 Conclusions and Further Issues

In this paper we compared four methods for Boolean matrix factorization by testing them on seven real datasets mainly taken from UCI Machine Learning Repository [1].

The results of experiments indicate that there is not a single method that performs the best for all datasets. In order to obtain a decomposition with a good coverage quality one has to experiment with different methods (and if possible, different combinations of their parameters). Moreover, the relative performances of methods cannot be, in general, predicted on the basis of simple features as are size and density of datasets. However, we do not claim that partial predictions cannot be made. For example, in a limited scope of our experiments BMDP performed, for a small number of factors, significantly better than the remaining three methods whenever the input dataset had a high density.

Another example would be the fact that Algorithm 1 and Algorithm 2 are always able to provide an exact decomposition (due to the universality of formal concepts as factors, see [4]) while the remaining methods do not have this ability in general. In other words, the coverage quality corresponding to the Boolean rank of an arbitrary matrix for Algorithms 1 and 2 is equal to 1. While, in most of the examined datasets, this coverage quality for Asso and BMDP is strictly smaller than 1.

Future research includes the following topics:

– A more detailed study of the two types of errors, E_o and E_u, made by the BMF algorithms. This seems to be an interesting, yet not epored characteristics. Note that both Algorithm 1 nor 2 make zero E_o error, yet they perform very well. This indicates a potential to make them perform still better by allowing for some E_o error in an appropriate way.
– Experimental as well as analytical comparison of computational complexity of BMF algorithms.
– In [2,3], the BMF problem is extended to ordinal data matrices and the method from [4] is extended to this type of data. It remains to be seen how the ideas behind other existing algorithms may be extended to this important type of data.
– Develop further applications of BMF, particularly in fields such as psychology or biosciences where Boolean data are abundant.

References

1. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences (2007). http://www.ics.uci.edu/~mlearn/MLRepository.html
2. Belohlavek, R.: Optimal decompositions of matrices with entries from residuated lattices. J. Logic Comput., 7 September 2011. doi:10.1093/logcom/exr023
3. Belohlavek, R., Vychodil, V.: Factor analysis of incidence data via novel decomposition of matrices. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548, pp. 83–97. Springer, Heidelberg (2009)
4. Belohlavek, R., Vychodil, V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. J. Comput. Syst. Sci. **76**(1), 3–20 (2010)
5. Berry, M.W., Browne, M., Langville, A.N., Pauca, V.P., Plemmons, R.J.: Algorithms and applications for approximate nonnegative matrix factorization. Comput. Stat. Data Anal. **52**, 155–173 (2007)
6. Cudeck, R., MacCallum, R.C. (eds.): Factor Analysis at 100: Historical Developments and Future Directions. Lawrence Erlbaum Associates Inc., Hillsdale (2007)

7. Fortelius, M., et al.: Neogene of the old world database of fossil mammals (NOW) (2003). http://www.helsinki.fi/science/now/
8. Frolov, A.A., Húsek, D., Polyakov, P.A.: Boolean factor analysis by Hopfield-like autoassociative memory. IEEE Trans. Neural Networks **18**(3), 698–707 (2007)
9. Ganter, B., Wille, R.: Formal Concept Analysis. Mathematical Foundations. Springer, Berlin (1999)
10. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. In: Suzuki, E., Arikawa, S. (eds.) DS 2004. LNCS (LNAI), vol. 3245, pp. 278–289. Springer, Heidelberg (2004)
11. Golub, G.A., Van Loan, C.F.: Matrix Computations, 3rd edn. The Johns Hopkins University Press, Baltimore (1995)
12. Harman, H.H.: Modern Factor Analysis, 2nd edn. The Univ. Chicago Press, Chicago (1970)
13. Kim, K.H.: Boolean Matrix Theory and Applications. M. Dekker, New York (1982)
14. Lee, D., Seung, H.: Learning parts of objects by non-negative matrix factorization. Nature **401**, 788–791 (1999)
15. Leeuw, J.D.: Principal component analysis of binary data. Application to roll-call analysis (2003). http://gifi.stat.ucla.edu
16. Lu, H., Vaidya, J., Atluri, V.: Optimal Boolean matrix decomposition: application to role engineering. In: Proceedings of IEEE ICDE 2008, pp. 297–306 (2008)
17. McDonald, R.P.: Factor Analysis and Related Methods. Lawrence Erlbaum Associates Inc., McHorney (1985)
18. Mickey, M.R., Mundle, P., Engelman, L.: Boolean factor analysis. In: Dixon, W.J. (ed.) BMDP Statistical Software Manual, vol. 2, pp. 849–860. University of California Press, Berkeley (1990). http://www.statistical-solutions-software.com/products-page/bmdp-statistical-software/
19. Miettinen, P.: Sparse Boolean matrix factorizations. In: Proceedings of 10th IEEE International Conference on Data Minig (ICDM2010), pp. 935–940 (2010)
20. Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. IEEE Trans. Knowl. Data Eng. **20**(10), 1348–1362 (2008). preliminary version in PKDD 2006, pp. 335–346
21. Monson, D.S., Pullman, J.N.: A survey of clique and biclique coverings and factorizations of (0,1)-matrices. Bull. ICA **14**, 17–86 (1995)
22. Myllykangas, S., Himberg, J., Böhling, T., Nagy, B., Hollmén, J., Knuutila, S.: DNA copy number amplification profiling of human neoplasms. Oncogene **25**(55), 7324–7332 (2006)
23. Nau, D.S.: Specificity covering: immunological and other applications, computational complexity and other mathematical properties, and a computer program. A.M. Thesis, Technical report CS-1976-7, Computer Sci. Dept., Duke Univ., Durham, N.C. (1976)
24. Nau, D.S., Markowsky, G., Woodbury, M.A., Amos, D.B.: A mathematical analysis of human leukocyte antigen serology. Math. Biosci. **40**, 243–270 (1978)
25. Outrata, J.: Boolean factor analysis for data preprocessing in machine learning. In: Proceedins of ICML 2010, Washington, D.C., USA, pp. 899–902 (2010)
26. Orlitsky, S.A.: Semi-parametric exponential family PCA. In: Saul, L.K., et al. (eds.) Advances in Neural Information Processing Systems 17. MIT Press, Cambridge (2005). http://books.nips.cc/papers/files/nips17/NIPS2004_0152.pdf
27. Seppänen, J.K., Bingham, E., Mannila, H.: A simple algorithm for topic identification in 0–1 data. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 423–434. Springer, Heidelberg (2003)

28. Schein, A., Saul, L., Ungar, L.: A generalized linear model for principal component analysis of binary data. In: Proceedings of International Workshop on Artificial Intelligence and Statistics, pp. 14–21 (2003)
29. Stockmeyer, L.J.: The set basis problem is NP-complete. IBM Research Report RC5431, Yorktown Heights, NY (1975)
30. Tang, F., Tao, H.: Binary principal component analysis. In: Proceedings of British Machine Vision Conference 2006, pp. 377–386 (2006)
31. Tatti, N., Mielikäinen, T., Gionis, A., Mannila, H.: What is the dimension of your binary data? In: The 2006 IEEE Conference on Data Mining (ICDM 2006), pp. 603–612. IEEE Computer Society (2006)
32. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: finding a minimal descriptive set of roles. In: ACM Symposium on Access Control Models and Technologies, pp. 175–184, June 2007
33. Zivkovic, Z., Verbeek, J.: Transformation invariant component analysis for binary images. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), vol. 1, pp. 254–259 (2006)

A Rough Fuzzy Perspective to Dimensionality Reduction

Alessio Ferone(✉) and Alfredo Petrosino

Department of Science and Technology, University of Naples Parthenope, Centro Direzionale Isola C4, 80143 Naples, Italy
{alessio.ferone,alfredo.petrosino}@uniparthenope.it

Abstract. Rough set theory and fuzzy logic are mathematical frameworks for granular computing forming a theoretical basis for the treatment of uncertainty in many real–world problems. The focus of rough set theory is on the ambiguity caused by limited discernibility of objects in the domain of discourse; granules are formed as objects and are drawn together by the limited discernibility among them. On the other hand, membership functions of fuzzy sets enables efficient handling of overlapping classes. The hybrid notion of rough fuzzy sets comes from the combination of these two models of uncertainty and helps to exploit, at the same time, properties like coarseness and vagueness. We describe a model of the hybridization of rough and fuzzy sets, that allows for further refinements of rough fuzzy sets and show its application to the task of unsupervised feature selection.

Keywords: Rough fuzzy sets · Modelling hierarchies · Unsupervised feature selection

1 Introduction

Feature selection concerns the selection of the most predictive input attributes with respect to a given outcome, with application in tasks that involve large number of features which would be difficult to process. The main difference with other dimensionality reduction methods, is that selected features preserve the original meaning of the features after reduction.

In the recent years, granular computing has been extensively employed for feature selection. It is based on the concept of information granule, that is a collection of similar objects which can be considered indistinguishable. Partition of an universe into granules offers a coarse view of the universe where concepts, represented as subsets, can be approximated by means of granules. In this framework, rough set theory can be regarded to as a family of methodologies and techniques that make use of granules [26,27]. The focus of rough set theory is on the ambiguity caused by limited discernibility of objects in the domain of discourse. Granules are formed as objects and are drawn together by the limited discernibility among them, without any apriori information needed. Granulation is of particular interest when a problem involves incomplete, uncertain or vague information. In such cases, precise solutions can be difficult to obtain and hence the use of techniques based on granules can lead to a simplification of the problem at hand.

The rough set ideology of using only the supplied data and no other information has many benefits in feature selection, although the requirement that all data has to be discrete imposes some limitations. In order to overcome these limitations, multivalued logic can be applied to handle uncertainty and vagueness present in information system, the most visible of which is the theory of fuzzy sets [41]. In this framework, uncertainty is modelled by means of functions that define the degree of belongingness of an object to a given concept. Hence membership functions of fuzzy sets enable efficient handling of overlapping classes.

The hybrid notion of rough fuzzy sets comes from the combination of these two models of uncertainty to exploit, at the same time, properties like coarseness, by handling rough sets [26], and vagueness, by handling fuzzy sets [41]. In this combined framework, rough sets embody the idea of indiscernibility between objects in a set, while fuzzy sets model the ill-definition of the boundary of a subclass of this set.

Nevertheless, some considerations are in order. Classical rough set theory is defined over a given partition, although several equivalence relations, and hence partitions, can be defined over the universe of discourse. Different partitions correspond to a coarser or finer view of the universe, because of different information granules, thus leading to coarser or finer definition of the concept to be provided. Then a substantial interest arises about the possibility of exploiting different partitions and, possibly, rough sets of higher order to reduce the dimensionality of data. This leads to a choice of the partition that represents the data in the best manner. In order to exploit different partitions, we propose to refine them in a hierarchical manner, so that partitions at each level of the hierarchy retain all the important information contained into the partitions of the lower levels. The operation employed to perform the hierarchical refinement is called *Rough–Fuzzy product* ($\mathcal{RF}$-product).

The hybridization of rough and fuzzy sets reported here has been observed to possess a viable and effective solution in feature selection. The model exhibits a certain advantage of having a new operator to compose rough fuzzy sets that is able to produce a sequence of composition of rough fuzzy sets in a hierarchical manner. Theory of $\mathcal{RF}$-product along with results and comparisons with other techniques are reported in the context of unsupervised feature selection.

The article is organized as follows. In Sect. 2 the literature about feature selection using rough and fuzzy theories is reviewed. In Sect. 3 rough–fuzzy sets are introduced along with the rough–fuzzy product operation, while in Sect. 4 their application to feature selection is explained. Section 5 presents the experimental results and Sect. 6 concludes the paper.

2 Related Works

As dimensions increase, the amount of data needs to increase as well, otherwise there would not be enough points to perform a useful analysis. This is the so-called curse of dimensionality [2]. In order to avoid spurious patterns that would lower the performance, most techniques employ some kind of dimensionality reduction.

Many problems in machine learning involve high dimensional descriptions of input features and therefore much research has been carried out on dimensionality reduction [7].

However, existing approaches tend to destroy the underlying semantic of the features [8] or require apriori information about the data [21]. In order to overcome these limitations, rough set theory is a technique that can reduce dimensionality using only information contained into the dataset, preserving at the same time the semantic of the features. Rough set theory can be used as such a tool to discover data dependencies and to reduce the number of attributes contained in a dataset [26].

In particular, the use of rough set theory to achieve dimensionality reduction has been proved to be a successful approach due to the following aspects:

- only the concepts embedded in data are analysed
- no apriori information about the data is required
- minimal knowledge representation is found

Given a dataset with discretized attribute values, rough set theory finds the most informative subset of the original attributes. This subset is called reduct.

Recently, researchers have focused their attention on reduct and classification algorithms based on rough sets [31] Finding all the reducts has been proved to be NP-Hard [19], but in many applications it is sufficient to compute only one reduct, that is the best reduct with respect to a given cost criterion associated with the selected attributes.

Nowadays many implementations of feature selection and classification techniques based on rough set theory are available [17].

Khoo et al. [18] proposed a novel approach for classification of inconsistent information systems, achieved by combining rough set theory and statistic inductive learning. Authors presented a rough set-based classification system that for each possible rule generated, was able to provide an estimation of the expected classification reliability. The proposed technique was compared with the other rule techniques (ID3 and LERS).

Bakar et al. [1] presented an algorithm for finding minimum size reducts based on rough set theory and binary integer programming. The idea is to transform a granule, obtained from a decision system, into a BIP model.

In [13] authors employ rough set theory in order to construct an ensemble of classifiers. The ensemble was theoretically formulated within the rough set theory framework and implemented by using set-oriented database operations. The proposed approach was used to compute a set of reducts comprising the minimum set of attribute required for the decision categories. For each reduct, a reduct table was generated and a rule induction algorithm was used to compute the maximal generalized rules for each reduct table. Finally, a set of classifiers were trained based on the corresponding reducts.

In [30], Questier et al. described how rough set theory can be employed to construct reducts as in a supervised approach but to reduce the number of features in unsupervised hierarchical clustering. The Wallace measure has been used to compare the results obtained on the original data set and those obtained on the reduced data set.

Swiniarski and Skowron [32] presented an application of rough set method for feature selection in pattern recognition. The proposed algorithm is based on the application of rough set theory to the result of principal components analysis (PCA). Experiments have been performed using neural networks for face and mammogram recognition.

Rough set-based methods have proved their effectiveness also in conjunction with neural network classifiers. In [10] Grzymala-Busse introduced a new algorithm called Modified Learning Examples Module Version 2 (MLEM2) able to induce rules from data with both symbolic and numerical attributes with missing attribute values. The algorithm was compared with LEM2, MODLEM Laplace and entropy-based algorithm, performing better in terms of number of rules as well as accuracy.

Hu et al. in [14] proposed a new rough sets model and two algorithms for computing core and reduct. These algorithms were applied in a real-life application with very large data sets proving to be efficient and scalable compared with traditional rough set models.

In [4] authors proposed a new definition of parametrization reduction for soft sets and compared it with reduction based on rough set theory. By employing this new definition, they improved the results of a soft set in a decision-making problem.

Thangavel et al. [33] proposed an algorithm to deal with data containing only input information (condition attributes) without decision (class attribute). K-Means algorithm was applied to cluster the given information system for different values of K and a decision table computed using the clustered data as decision attribute. Quickreduct and Variable Precision Rough Set (VPRS) algorithms were applied for selecting features. Different data sets from the UCI machine learning repository were used to evaluate the performance of the proposed approach, yielding good results when used in combination with VPRS.

Thangavel et al. [34] proposed a Modified Quickreduct algorithm and compared its performance with different reduct algorithms such as Quickreduct and VPRS on data sets from UCI repository. The proposed algorithm allowing both vertical reduct (feature selection) and horizontal reduct (object selection) generated minimal reducts with respect to Quickreduct and VPRS.

Authors in [35] proposed an Accelerated Quickreduct algorithm whose performance was compared with the original Quickreduct algorithm. The experiments, carried out on datasets available in UCI repository, made clear that the Accelerated Quickreduct produced the minimal reduct and the rules induced through C4.5 algorithm revealed that the Accelerated Quickreduct was performing well.

A major drawback when using rough set theory is represented by real value attributes, because it is not possible to say whether two attribute values are similar and to what extent they are indiscernible. A possible solution to this problem consists in discretizing the real valued attributes in order to obtain a dataset composed only by crisp values. This preprocessing step is not always adequate, being the source of potential information loss.

Fuzzy sets provide a framework to handle real value data effectively, by allowing values to belong to more than one class with different degrees of membership, and hence handling vagueness present in data.

In an hybridized approach, rough set theory allows to obtain a linguistic description whereas fuzzy set theory allows to generate numerical values starting from its linguistic description.

Jensen and Shen [17] reviewed semantics–preserving dimensionality reduction technique using fuzzy–rough set approaches. The result showed that rough set methods were non able to deal with real–valued attributes effectively and hence the need of rough and fuzzy hybridized approaches. To this aim authors present the Quickreduct algorithm which exploits at the same time both rough and fuzzy sets theories.

In [37], Tsai et al. proposed a new fuzzification technique called Modified Minimization Entropy Principle Algorithm (MMEPA) to construct membership functions of linguistic variables. The proposed algorithm was combined with VPRS yielding an entropy-based fuzzy–rough classification approach.

Authors in [12] proposed a method which combined VPRS and fuzzy set theory to produce a set of fuzzy certain and fuzzy possible rules from quantitative data given an uncertainty and misclassification tolerance degree. Rough set theory was employed to map each quantitative value into a fuzzy set of linguistic terms and then to compute the fuzzy lower and upper approximations. The rules obtained by the two approximations sets were used to classify unknown objects.

Thangavel and Pethalakshmi [36] proposed an Improved Quickreduct algorithm based on rough and fuzzy sets. In the proposed algorithm, the attributes were first normalized and the procedure to derive the degree of dependency of each attribute has been modified accordingly. Experimental results showed that the Improved Quickreduct produces minimal reduct.

In [38] a novel concept of attributes reduction with fuzzy rough sets is proposed and an algorithm using discernibility matrix to compute all the reducts is developed. A solid mathematical foundation is set up for attributes reduction with fuzzy rough sets and a detailed comparison with the Quickreuct algorithm is also presented. Experimental results show that the proposed algorithm is feasible and valid.

Authors in [3] generalize the classical rough set framework for attribute reduction within the context of fuzzy rough set theory, based on the notion of fuzzy decision reducts. The concept of a fuzzy decision reduct is introduced, as the weighted version of its crisp version. Its role is to assign to each attribute subset a measure of the predictive power with respect to the original decision system. Authors consider also alternative ways of defining fuzzy decision reducts. Experimental results showed the potential of the proposed approach.

Jensen and Shen in [15] proposed an extension of the fuzzy–rough feature selection algorithm, based on interval–valued fuzzy sets, in order to face the problem of missing values. In particular, by exploiting interval–valued fuzzy–rough sets, a new feature selection algorithm is developed that not only handles missing values, but also alleviates the problem of defining overly–specific type–1 fuzzy similarity relations.

In [16] three new robust approaches to fuzzy–rough feature selection based on fuzzy similarity relations are proposed. In particular, a fuzzy extension to crisp discernibility matrices is proposed and employed for experimentation, showing that the methods produce small reduct while preserving classification accuracy.

Parthaláin et al. [24] examine a rough set based feature selection technique which exploits information extracted from the lower approximation, from the boundary region and the distance of objects in the boundary region from the lower approximation. This information allows to obtain smaller subset if compared to those obtained using the dependency function alone. The proposed approach demonstrates that information extracted from the boundary region is useful for feature selection.

In [20] a novel fuzzy-rough sets based feature selection method is presented by maximizing the relevance and minimizing the redundancy of the selected features. The fuzzy equivalence partition matrix is introduced in order to compute many f-information measures and a novel entropy for fuzzy approximation spaces is proposed to measure the relevance and redundancy of features. The f-information measures have been shown to be effective for selecting non-redundant and relevant features. Also some fuzzy-rough set based quantitative indexes are introduced for evaluating the performance of the proposed method.

Supervised feature selection methods evaluate subsets of features using an objective function in order to select only those features related to the the decision classes. However, in many applications, class labels are not available or incomplete, and unsupervised feature selection approaches are needed. Approaches to unsupervised feature selection can be divided in two broad classes: those that maximize clustering performance with respect to an index function [6,23], and those that select features based on their relevance. The main idea of the latter methods, is that features with little or no information with respect to the remaining features are redundant and can be eliminated [5,11,22]. The work presented in [25] is based on fuzzy-rough sets and, in particular, employs a fuzzy-rough discernibility measure to compute the discernibility between a single feature and a subset of other features. If the single feature can be discerned by the subset of the other features, than it is considered redundant and removed from the feature set. Features are removed until no further inter-dependency can be found.

In [39] authors propose a new unsupervised quick reduct algorithm based on rough set theory. The proposed algorithm is based on a new definition of positive region for unsupervised subset evaluation measure using rough set theory. The evaluation of degree of dependency value for a features subset leads to each conditional attribute and evaluate mean of dependency values for all conditional attributes.

3 Rough-Fuzzy Sets

Let us start from the definition of a rough fuzzy set given by Dubois and Prade [9]. Let U be the universe of discourse, X a fuzzy subset of U, such that $\mu_X(u)$ represents the fuzzy membership function of X over U, and R an equivalence relation that induces the partition $U/R = \{Y_1, \ldots, Y_p\}$ (from now on denoted as $\mathcal{Y}$) over U in p disjoint sets, i.e. $Y_i \bigcap Y_j = \emptyset \; \forall i, j = 1, \ldots, p$ and $\bigcup_{i=1}^{p} Y_i = U$.

Considering the lower and upper approximations of the fuzzy subset X as, respectively, the infimum and the supremum of the membership functions of the elements of a class Y_i to the fuzzy set X [29], a rough-fuzzy set can be defined as a triple

$$RF_X = (\mathcal{Y}, \mathcal{I}, \mathcal{S}) \tag{1}$$

where $\mathcal{Y} = \{Y_1, \ldots, Y_p\}$ is a partition of U in p disjoint subsets $Y_1, \ldots, Y_p$, and $\mathcal{I}, \mathcal{S}$ are mappings of kind $U \rightarrow [0, 1]$ such that $\forall u \in U$,

$$\mathcal{I}(u) = \sum_{i=1}^{p} \underline{\nu_i} \times \mu_{Y_i}(u) \tag{2}$$

$$\mathcal{S}(u) = \sum_{i=1}^{p} \overline{\nu_i} \times \mu_{Y_i}(u) \tag{3}$$

where

$$\underline{\nu}_i = \inf\{\mu_X(u) | u \in Y_i\} \tag{4}$$

$$\overline{\nu}_i = \sup\{\mu_X(u) | u \in Y_i\} \tag{5}$$

$\mathcal{Y}$ and μ uniquely define a rough-fuzzy set.

In the proposed framework, different partitions, possibly obtained by applying different equivalent relations, are refined in a hierarchical manner, so that partitions at each level of the hierarchy retain all the important information contained into the partitions of the lower levels. The operation employed to perform the hierarchical refinement is called *Rough–Fuzzy product* ($\mathcal{RF}$-product) and is defined by:

Definition 1. *Let $RF^i = (\mathcal{Y}^i, \mathcal{I}^i, \mathcal{S}^i)$ and $RF^j = (\mathcal{Y}^j, \mathcal{I}^j, \mathcal{S}^j)$ be two rough fuzzy sets defined, respectively, over partitions $\mathcal{Y}^i = (Y_1^i, \ldots, Y_p^i)$ and $\mathcal{Y}^j = (Y_1^j, \ldots, Y_p^j)$ with $\mathcal{I}^i$ (resp. $\mathcal{I}^j$) and $\mathcal{S}^i$ (resp. $\mathcal{S}^j$) indicating the measures expressed in Eqs. (2) and (3). The $\mathcal{RF}$–product between two rough-fuzzy sets, denoted by $\otimes$, is defined as a new rough fuzzy set*

$$RF^{i,j} = RF^i \otimes RF^j = (\mathcal{Y}^{i,j}, \mathcal{I}^{i,j}, \mathcal{S}^{i,j}) \tag{6}$$

where $\mathcal{Y}^{i,j} = (Y_1^{i,j}, \ldots, Y_{2p-1}^{i,j})$ is a new partition whose equivalence classes are

$$Y_k^{ij} = \begin{cases} \bigcup\limits_{\substack{s=1\\q=h}}^{\substack{s=h\\q=1}} Y_q^i \cap Y_s^j & h = k, \qquad k \le p \\ \\ \bigcup\limits_{\substack{s=h\\q=p}}^{\substack{s=p\\q=h}} Y_q^i \cap Y_s^j & h = k-p+1, \qquad k > p \end{cases} \tag{7}$$

and $\mathcal{I}^{i,j}$ *and* $\mathcal{S}^{i,j}$ *are*

$$\mathcal{I}^{i,j}(u) = \sum_{k=1}^{2p-1} \underline{\nu}_k^{i,j} \times \mu_k^{i,j}(u) \tag{8}$$

$$\mathcal{S}^{i,j}(u) = \sum_{k=1}^{2p-1} \overline{\nu}_k^{i,j} \times \mu_k^{i,j}(u) \tag{9}$$

and

$$\underline{\nu}_k^{ij} = \begin{cases} \sup\limits_{\substack{s=1,\ldots,h\\q=h,\ldots,1}} \{\underline{\nu}_q^i, \underline{\nu}_s^i\} & h = k, \qquad k \le p \\ \\ \sup\limits_{\substack{s=h,\ldots,p\\q=p,\ldots,h}} \{\underline{\nu}_q^i, \underline{\nu}_s^i\} & h = k-p+1, \qquad k > p \end{cases} \tag{10}$$

$$\overline{\nu}_k^{ij} = \begin{cases} \inf\limits_{\substack{s=1,\ldots,h\\q=h,\ldots,1}} \{\overline{\nu}_q^i, \overline{\nu}_s^i\} & h = k, \qquad k \le p \\ \\ \inf\limits_{\substack{s=h,\ldots,p\\q=p,\ldots,h}} \{\overline{\nu}_q^i, \overline{\nu}_s^i\} & h = k-p+1, \qquad k > p \end{cases} \tag{11}$$

4 Rough-Fuzzy Product Feature Selection

In this section we describe how the rough–fuzzy product can be exploited in order to find distinctive features in an unsupervesided way. In particular, the proposed approach is composed by two steps

1. feature granularization, which consists in partitioning the data considering each single feature and building a rough–fuzzy set for each partition
2. feature selection, which consists in combining rough–fuzzy sets by means of rough–fuzzy product and selecting the most distinctive features

4.1 Feature Granularization

The feature granularization step is based on the principle of justifiable granularity [28], that is concerned with the formation of a meaningful information granule Ω based on some experimental evidence of scalar numeric data, $D = x_1, x_2, \ldots, x_N$. Such construct has to respect two requirements:

1. The numeric evidence accumulated within the bounds of Ω has to be as high as possible, i.e. the existence of the information granule is well motivated, or justified, by the experimental data.
2. The information granule should be as specific as possible meaning that it represents a well-defined semantics, i.e. Ω has to be as specific as possible.

Let us consider Ω as an interval to be constructed. In the simplest case, the first requirement is quantified by counting the number of data falling within the bounds of Ω, specifically we consider

$$f_1(card(X_k \in \Omega)) = card(X_k \in \Omega) \tag{12}$$

The specificity of the information granule can be quantified by taking into account its size. The length of the interval Ω can be considered as a measure of specificity, in particular we consider

$$f_2(length(\Omega)) = exp(-\alpha|a - m|) \tag{13}$$

The lower the value of $f2(length(\Omega))$, the higher the specificity is.

In order to construct the interval information granules, we start with the determination of the numeric representative of the set of data D. A sound representative is its median, $med(D)$, as it is a robust estimator of the sample and typically comes as one of the elements of D. An information granule Ω is formed by forming its lower and upper bounds, denoted by a and b, respectively.

The length of Ω, which quantifies the specificity of the information granule, is given now as $|a-m|$, where $m = med(D)$. More generally, we employ $f2(|a-m|)$ where $f2$ is a nonincreasing function, i.e. $f_2(length(\Omega)) = exp(-\alpha|a-m|)$, where α is a positive parameter offering some flexibility in the produced information granule. The optimal granularization is obtained by maximizing the sum over all the granules Ω_i

$$\sum_i V(\Omega_i) \tag{14}$$

where

$$V(\Omega_i) = f_1(card(X_k \in \Omega_i)) * f_2(length(\Omega_i)) \tag{15}$$

As the requirements of experimental evidence and specificity are in conflict, we consider the maximization of the product $V = f_1 * f_2$.

4.2 Feature Selection

The same procedure is applied independently to each feature of the dataset, thus yielding many partitions of the same dataset. As explained before, for each partition it is possible to define a rough–fuzzy set which can be composed by means of the rough–fuzzy product. Let $RF^i = (\mathcal{Y}^i, \mathcal{I}^i, \mathcal{S}^i)$ and $RF^j = (\mathcal{Y}^j, \mathcal{I}^j, \mathcal{S}^j)$ be two rough–fuzzy sets relative to feature i and j, and let $RF^{i,j} = RF^i \otimes RF^j = (\mathcal{Y}^{i,j}, \mathcal{I}^{i,j}, \mathcal{S}^{i,j})$ be the rough–fuzzy set obtained by applying the rough–fuzzy product to these rough–fuzzy sets. In order to evaluate the goodness of the newly formed rough–fuzzy set with respect the operands, we propose to exploit again the principle of justifiable granularity, where for each granule $Y_k^{i,j}$ of the new rough–fuzzy set

$$f_1(card(Y_k^{i,j})) = card(Y_k^{i,j}) \tag{16}$$

represents the experimental evidence and

$$f_2(Y_k^{i,j}) = exp(-\beta|\underline{\nu}_k^{i,j} - \overline{\nu}_k^{i,j}|) \tag{17}$$

represents the spread with respect to the membership degrees. The optimal rough–fuzzy set is obtained by maximizing

$$\sum_{k=1}^{p} V(Y_k^{i,j}) \tag{18}$$

where

$$V(Y_k^{i,j}) = card(Y_k^{i,j}) * exp(-\beta|\underline{\nu}_k^{i,j} - \overline{\nu}_k^{i,j}|) \tag{19}$$

The Rough-Fuzzy Product Feature Selection Algorithm is sketched in Algorithm 1. First each feature is granularized by maximizing Eq. 15 and a rough–fuzzy set is constructed as defined in Eq. 1 (lines 2–5). Second the couple of features that maximize Eq. 19 is found by applying the rough–fuzzy product in Eq. 6 (lines 8–15). The remaining features, are added one at time only if the rough–fuzzy set obtained by rough–fuzzy producting the rough–fuzzy sets of the new feature and the already selected features, leads to a better solution with respect to Eq. 19.

5 Experimental Results

In this section, experimental results for the proposed approach are presented. The method is compared with some supervised and unsupervised methods. The comparison with the supervised methods is included to show that despite missing or incomplete labels, RFPFS can effectively reduce dimensionality and discover useful subsets of features. The experimental setup consists of three steps: (1) feature selection, (2) dataset reduction by retaining selected features, (3) classifier learning. Note that the class label have been removed before applying

Algorithm 1. RFPFS - Rough Fuzzy Product Feature Selection

```
1: $F = \{\text{set of features}\}$
2: for all $c \in F$ do
3:     Granularization of c
4:     Rough–Fuzzy Set of c
5: end for
6: RFPD={$\emptyset$}
7: $V_{\max} = 0$
8: for all $i, j \in F$ do
9:     $RF^{i,j} = RF^i \otimes RF^j$
10:    $V(RF^{i,j}) = \sum_{k=1}^{p} V(Y_k^{i,j})$
11:    if $V(RF^{i,j}) > V_{\max}$ then
12:        $V_{\max} = V(RF^{i,j})$
13:        $RFPD = \{i, j\}$
14:    end if
15: end for
16: for all $c \in F$ and $c \notin RFPD$ do
17:    $RF^{RFPD \cup c} = RF^{RFPD} \otimes RF^c$
18:    $V(RF^{RFPD \cup c}) = \sum_{k=1}^{p} V(Y_k^{RFPD \cup c})$
19:    if $V(RF^{RFPD \cup c}) > V_{\max}$ then
20:        $V_{\max} = V(RF^{RFPD \cup c})$
21:        $RFPD = RFPD \cup \{c\}$
22:    end if
23: end for
```

RFPFS algorithm. The classifier used in tests is the J48 classifier that creates decision trees by choosing the most informative features via an entropy measure, and recursively partitions the data into subtables based on their values. Each node in the tree represents a feature with branches from a node representing the alternative values this feature can take according to the current subtable. Partitioning stops when all data items in the subtable have the same classification.

The first test has been performed on three datasets from the UCI repository, namely Wine (178 instances and 13 features), Wisconsin (569 instances and 32 features), and Sonar (208 instances and 60 features). From Table 1 it is possible to note how the proposed algorithm selects approximately half of the features (1/3 in the Sonar dataset) still obtaining good classification accuracy with respect to the unreduced dataset.

Table 1. Results on UCI datasets.

	Unreduced	RFPFS (No. of selected features)
Wine	94.41	93.80 (6)
Wisconsin	72.46	73.62 (14)
Sonar	93.86	95.03 (22)

Table 2. RFPFS Vs. Unsupervised features selection algorithms.

	Unreduced	UFRFS	B–UFRFS	D–UFRFS	RFPFS
Wine	94.41 (13)	79.74 (7)	81.99 (7)	79.74 (6)	93.80 (6)
Water	83.08 (38)	81.54 (7)	80.51 (7)	81.54 (7)	80.30 (4)

Table 3. RFPFS Vs. Supervised features selection algorithms.

	Unreduced	CFS	Consis	FRFS	B–FRFS	D–FRFS	RFPFS
Wine	94.41 (13)	94.41 (11)	97.10 (5)	94.97 (5)	96.08 (5)	94.41 (5)	93.80 (6)
Water	83.08 (38)	81.54 (11)	81.02 (11)	79.49 (6)	80.26 (6)	80.77 (6)	80.30 (4)

In the second test, the proposed method has been compared to some supervised (correlation-based (CFS) [11], consistency-based [40], fuzzy-rough lower approximation-based (FRFS) [16], boundary region-based (B-FRFS) [16], discernibility-based (D-FRFS) [16]) and unsupervised (fuzzy-rough lower approximation-based (UFRFS) [25], unsupervised boundary region-based (B-UFRFS) [25] and unsupervised discernibility-based (D-UFRFS) [25]) feature selection methods. The dataset used in this test are Wine as in the first test, and Water (390 instances and 38 features).

From Table 2 it is possible to see how RFPFS clearly outperforms the considered unsupervised approaches on the Wine dataset while on the Water dataset the results are comparable but with less features selected. The results are even more interesting if compared with those obtained by the supervised approaches, shown in Table 3. Even in this case the proposed method is comparable with the other approaches in terms of accuracy, but without considering the class labels.

6 Conclusions

This paper has presented a novel technique for unsupervised feature selection, based on the rough–fuzzy product operation already presented in the literature. The proposed approach is data-driven, and no user-defined thresholds or domain-related information is required. The experimental results show that the approach can reduce dataset dimensionality considerably whilst retaining useful features when class labels are unknown or missing.

References

1. Bakar, A.A., Sulaiman, M.N., Othman, M., Selamat, M.H.: Finding minimal reduct with binary integer programming in data mining. In: Proceedings of the TENCON, pp. 141–149 (2000)
2. Bellman, R.: Adaptive Control Processes: A Guided Tour. Princeton University Press, Princeton (1961)

3. Chanas, S., Kuchta, D.: Further remarks on the relation between rough and fuzzy sets. Fuzzy Sets Syst. **47**(3), 391–394 (1992)
4. Chen, D., Tsang, E.C.C., Yeung, D.S., Wang, X.: The parameterization reduction of soft sets and its applications. Int. J. Comput. Math. **49**, 757–763 (2005)
5. Das, S.K.: Feature selection with a linear dependence measure. IEEE Trans. Comput. **20**, 1106–1109 (1971)
6. Dash, M., Liu, H.: Unsupervised feature selection. In: Proceedings of the Pacific and Asia Conference on Knowledge Discovery and Data Mining, pp. 110–121 (2000)
7. Dash, M., Liu, H.: Feature selection for classification. Intell. Data Anal. **1**(3), 131–156 (1997)
8. Devijver, P., Kittler, J.: Pattern Recognition: A Statistical Approach. Prentice Hall, London (1982)
9. Dubois, D., Prade, H.: Rough fuzzy sets and fuzzy rough sets. Int. J. Gen Syst **17**(2–3), 191–209 (1990)
10. Grzymala-Busse, J.W.: MLEM2-discretization during rule induction. In: Procedings of IIPWM (2003)
11. Hall, M.A.: Correlation-based feature selection for discrete and numeric class machine learning. In: Proceedings of the 17th International Conference on Machine Learning, pp. 359–366 (2000)
12. Hong, T.-P., Tseng, L.-H., Wang, S.-L.: Learning rules from incomplete training examples by rough sets. Expert Syst. Appl. **22**, 285–293 (2002)
13. Hu, X.: Using rough sets theory and database operations to construct a good ensemble of classifiers for data mining applications. In: Proceedings of ICDM, pp. 233–240 (2001)
14. Hu, X., Lin, T.Y., Jianchao, J.: A new rough sets model based on database systems. Fundamenta Informaticae **20**, 1–18 (2004)
15. Jensen, R., Shen, Q.: Interval-valued fuzzy-rough feature selection in datasets with missing values. In: IEEE International Conference on Fuzzy Systems, pp. 610–615 (2009)
16. Jensen, R., Shen, Q.: New approaches to fuzzy-rough feature selection. IEEE Trans. Fuzzy Syst. **17**(4), 824–838 (2009)
17. Jensen, R., Shen, Q.: Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches. IEEE Trans. Knowl. Data Eng. **16**(12), 1457–1471 (2004)
18. Khoo, L.P., Tor, S.B., Zhai, L.Y.: A rough set-based approach for classification and rule induction. Int. J. Adv. Manuf. Technol. **15**, 438–444 (1999)
19. Lin, T.Y., Cercone, N.: Rough sets and Data Mining: Analysis of Imprecise Data. Kluwer Academic Publishers, Boston (1997)
20. Maji, P., Pal, S.K.: Feature selection using f-information measures in fuzzy approximation spaces. IEEE Trans. Knowl. Data Eng. **22**(6), 854–867 (2010)
21. Mitchell, T.: Machine Learning. McGraw-Hill, Maidenhead (1997)
22. Mitra, P., Murthy, C.A., Pal, S.K.: Unsupervised feature selection using feature similarity. IEEE Trans. Pattern Anal. Mach. Intell. **24**(4), 1–13 (2002)
23. Pal, S.K., De, R.K., Basak, J.: Unsupervised feature evaluation: a neuro-fuzzy approach. IEEE Trans. Neural Network. **11**, 366–376 (2000)
24. Parthaláin, N.M., Shen, Q., Jensen, R.: A distance measure approach to exploring the rough set boundary region for attribute reduction. IEEE Trans. Knowl. Data Eng. **22**(3), 305–317 (2010)
25. Parthaláin, N.M., Jensen, R.: Measures for unsupervised fuzzy-rough feature selection. Int. J. Hybrid Intell. Syst. **7**(4), 249–259 (2010)

26. Pawlak, Z.: Rough sets. Int. J. Comput. Inform. Sci. **11**, 341–356 (1982)
27. Pawlak, Z.: Granularity of knowledge, indiscernibility and rough sets. In: Proceedings of IEEE International Conference on Fuzzy Systems, pp. 106–110 (1998)
28. Pedrycz, W., Gomide, F.: Fuzzy Systems Engineering: Toward Human-Centric Computing. Wiley, Hoboken (2007)
29. Petrosino, A., Ferone, A.: Feature discovery through hierarchies of rough fuzzy sets. In: Chen, S.-M., Pedrycz, W. (eds.) Granular Computing and Intelligent Systems: Design with Information Granules of Higher Order and Higher Type. Springer, Heidelberg (2011)
30. Questier, F., Rollier, I.A., Walczak, B., Massart, D.L.: Application of rough set theory to feature selection for unsupervised clustering. Chemometr. Intell. Lab. Syst. **63**, 55–167 (2002)
31. Shen, Q., Chouchoulas, A.: A modular approach to generating fuzzy rules with reduced attributes for the monitoring of complex systems. Eng. Appl. Artif. Intell. **13**(3), 263–278 (2002)
32. Swiniarski, R.W., Skowron, A.: Rough set methods in feature selection and recognition. Pattern Recogn. Lett. **24**, 833–849 (2003)
33. Thangavel, K., Shen, Q., Pethalakshmi, A.: Application of clustering for feature selection based on rough set theory approach. AIML J. **6**(1), 19–27 (2005)
34. Thangavel, K., Pethalakshmi, A., Jaganathan, P.: em A comparative analysis of feature selection algorithms based on rough set theory. Int. J. Soft Comput. **1**(4), 288–294 (2006)
35. Thangavel, K., Pethalakshmi, A.: Performance analysis of accelerated Quickreduct algorithm. In: Proceedings of International Conference on Computational Intelligence and Multimedia Applications, pp. 318–322 (2007)
36. Thangavel, K., Pethalakshmi, A.: Feature selection for medical database using rough system. Int. J. Artif. Intell. Mach. Learn. **6**, 11–17 (2005)
37. Tsai, Y.-C., Cheng, C.-H., Chang, J.-R.: Entropy-based fuzzy rough classification approach for extracting classification rules. Expert Syst. Appl. **31**(2), 436–443 (2006)
38. Tsang, E.C.C., Chen, D., Yeung, D.S., Wang, X.-Z., Lee, J.: Attributes reduction using fuzzy rough sets. IEEE Trans. Fuzzy Syst. **16**(5), 1130–1141 (2008)
39. Velayutham, C., Thangavel, K.: Unsupervised quick reduct algorithm using rough set theory. J. Electron. Sci. Technol. **9**(3), 193–201 (2011)
40. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools with Java Implementations. Morgan Kaufmann, San Francisco (2000)
41. Zadeh, L.: Fuzzy sets. Inf. Control **8**, 338–353 (1964)

Author Index