2nd MultiClust Workshop:
Discovering, Summarizing and Using Multiple Clusterings

WORKSHOP NOTES

Editors:
Emmanuel Müller
Stephan Günnemann
Ira Assent
Thomas Seidl

ECML PKDD 2011

EUROPEAN CONFERENCE ON MACHINE LEARNING
AND
PRINCIPLES AND PRACTICE OF KNOWLEDGE DISCOVERY IN DATABASES

2nd MultiClust Workshop:
Discovering, Summarizing and Using Multiple Clusterings

MultiClust'11

September 5, 2011

Athens, Greece

Editors:
Emmanuel Müller
Karlsruhe Institute of Technology, Germany
Stephan Günnemann
RWTH Aachen University, Germany
Ira Assent
Aarhus University, Denmark
Thomas Seidl
RWTH Aachen University, Germany

Preface

Traditional clustering algorithms identify just a single clustering of the data. However, today's complex and high dimensional data allow multiple interpretations for each data object, and thus, several valid groupings (multiple clustering solutions) can be detected. Recently, an emerging research direction, focusing on detecting, summarizing and using such multiple clustering solutions, has evolved out of this problem. This new clustering paradigm has attracted attention from many researchers and resulted in a number of important publications at leading data mining and machine learning conferences. Focusing on this novel paradigm, the MultiClust workshop attracts a variety of researchers working on different problem instances of multiple clustering solutions.

MultiClust 2011 establishes a venue for the growing community interested in multiple clustering solutions. As a platform for exchange of ideas, the workshop brings together researchers from well-established related areas, such as ensemble clustering, constraint-based clustering, frequent pattern mining, subspace mining and cluster exploration and visualization. The workshop covers aspects of these related fields and has its focus on the emerging cross-disciplinary topics. Overall, the workshop provides a venue for exploring state-of-the-art methods, presenting emerging work and discussing with active researchers in this field.

The technical program for this workshop includes seven peer-reviewed papers. They passed a competitive selection process ensuring high quality publications. Authors present a variety of aspects out of several research directions, and contribute to the emerging topic of multiple clustering solutions. We would like to highlight "Generating a Diverse Set of High-Quality Clusterings" by Jeff M. Phillips, Parasaran Raman and Suresh Venkatasubramanian. It has been selected as best contribution and receives the Best Paper Award from the MultiClust 2011 workshop.

Overall, the workshop demonstrates the strong interest from different research communities, and we are pleased to have some of the core researchers on the MultiClust 2011 program committee. We are particularly pleased to have two excellent speakers giving invited talks that provide an overview on challenges in related fields: Michael Houle (National Institute of Informatics, Japan) and Bart Goethals (University of Antwerp, Belgium) contribute with their cross-disciplinary research perspectives in feature selection and frequent itemset mining.

In the spirit of last year's workshop, the panel opens for a discussion of state-of-the-art, open challenges and visions for future research. It wraps up the workshop by summarizing several common challenges in different research directions, establishing novel research collaborations, and also providing a guideline for important topics to be addressed in following workshops.

We are grateful for the support of the ECML PKDD conference, assisting us in the workshop organization, and the MultiClust 2011 program committee for conducting thorough reviews of the submitted technical papers. We also wish to acknowledge the UMIC Research Centre at RWTH Aachen University in Germany for its support in making MultiClust 2011 possible.

Athens, September 2011

Emmanuel Müller
Stephan Günnemann
Ira Assent
Thomas Seidl

Workshop Organization

Workshop Chairs

Emmanuel Müller	Karlsruhe Institute of Technology, Germany
Stephan Günnemann	RWTH Aachen University, Germany
Ira Assent	Aarhus University, Denmark
Thomas Seidl	RWTH Aachen University, Germany

Program Committee

James Bailey	University of Melbourne, Australia
Carlotta Domeniconi	George Mason University, USA
Ines Färber	RWTH Aachen University, Germany
Vivekanand Gopalkrishnan	Nanyang Technological University, Singapore
Dimitrios Gunopulos	University of Athens, Greece
Michael Houle	National Institute of Informatics, Japan
Daniel Keim	University of Konstanz, Germany
Themis Palpanas	University of Trento, Italy
Magda Procopiuc	AT&T Labs, USA
Naren Ramakrishnan	Virginia Tech, USA
Jörg Sander	University of Alberta, Canada
Lyle Ungar	University of Pennsylvania, USA
Jilles Vreeken	University of Antwerp, Belgium
Arthur Zimek	University of Munich, Germany

Table of Contents

Invited Talks

Research Papers

Combinatorial Approaches to Clustering and Feature Selection

Michael E. Houle

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
http://www.nii.ac.jp/en/
meh@nii.ac.jp

Extended Abstract

1 Introduction

One of the most serious difficulties in the analysis of high-dimensional data sets involves the treatment of measures of similarity. Although similarity measures often retain some discriminative ability as the dimension increases, the similarity values themselves are often difficult to interpret. Methods for search, clustering and feature selection that perform quantitive tests of similarity values (as opposed to comparative tests) are particularly susceptible to this problem. This presentation will be concerned with combinatorial models of clustering based on shared neighbor information, and their application to feature selection, subspace clustering, and multiple clustering. The models assume a secondary, derived form of similarity measure based on the intersection properties of neighborhoods defined according to the original similarity measure. The use of secondary similarity has been recently shown to offer solutions that are more robust and more scalable with respect to the dimension of the data.

2 Secondary Similarity Measures

For similarity search and their applications, the distance measures commonly used in practice are known to be sensitive to local variations within the data distribution, as well as the number of data features involved (the dimension). These dependencies can severely limit the the efficiency and accuracy of the search, and ultimately the quality of the solution — a phenomenon often referred to as the *curse of dimensionality*. Generally speaking, as the number of data features increases, pairwise distance values tend to concentrate tightly about their mean, reducing the overall discriminative ability of the similarity measure. The effect occurs for a broad range of data distributions and similarity measures, and can be so pronounced as to cast doubt upon whether efficient nearest neighbor search is even achievable in higher dimensions [1]. However, when a data set is composed of many well-formed clusters, the concentration effect will typically be less severe across cluster boundaries, with distances from a cluster member

to other cluster members being relatively easy to distinguish from distances to non-members, especially when the clusters are well separated [1–3].

In general, any improvement in the discriminative ability of the similarity measure employed can be expected to yield improvements in the performances of solutions based on it. Some simple enhancement strategies involve the use of *shared neighbor* (SN) information, in which a *secondary* similarity between two points v and w is defined in terms of data objects in the common intersection of neighborhoods based at v and w, where the neighborhoods themselves are determined according to a supplied *primary* similarity measure. The primary measure can be any function that determines a well-defined ranking of the data objects relative to the query. Recent studies have shown that secondary similarity measures based on SN information are generally more robust in higher dimensions than their associated primary distance measures, since the neighborhoods of object pairs drawn from a common cluster tend to have significantly more items in common than to pairs drawn from different clusters [4, 5]. Furthermore, recent advances in approximate similarity search allow for neighborhood information to be generated accurately and efficiently for many practical applications [6, 7].

3 Multi-Source RSC Clustering

Shared-neighbor information has been used to guide clustering algorithms for almost 40 years [8–10]. However, early methods required that the neighborhood size k be fixed in advance by the user. The use of fixed values of k can introduce a very significant bias on the sizes and other characteristics of clusters that can be produced by the methods, in that they tend to favor the discovery of groups with size of roughly the same order as k.

In order to account for the effects of varying k, the Relevant-Set Correlation (RSC) model for clustering was proposed [11]. The RSC model provides a consistent and comprehensive framework for the assessment of cluster quality, based on the statistical significance of a form of correlation between the neighborhood sets of its members. More precisely, the model tests the significance of any grouping against the assumption that the neighborhoods contain zero information (that is, against the assumption that they were generated by means of random selection). The greater the extent to which the assumption is violated, the greater the significance of the grouping.

The RSC model quantifies the quality of cluster candidates of any arbitrary size (allowing the comparison of any two candidates regardless of their size), the degree of association between pairs of cluster candidates, and the degree of association between clusters and individual data items. An efficient greedy selection strategy, GreedyRSC, has been developed based on RSC, and was shown to be very competitive in practice [11]. It requires only two user-supplied parameters, describing the minimum acceptable cluster size, and the size of the maximum acceptable overlap between two clusters. Both of these parameters can be chosen in a natural way with no knowledge of the nature of the data or its distribution. The number of clusters is not specified by the user.

This presentation will be concerned with an extension of the RSC model to account for multiple sources of neighborhood information. Each of these sources is assumed to have its own similarity measure based on its own collection of data features (which may or may not contain features also used by other sources). Like the original RSC model, the extended model relies only on the neighborhood rankings produced according to the sources, and has no knowledge of the nature of the similarity measure or features involved.

The extension of RSC will be seen to have implications for subspace clustering and feature selection, as well as multiclustering. In particular, the discussion will include the following potential applications of the extended model:

- The significance of data sets can be simultaneously assessed with respect to object membership as well as the number of sources of neighborhood information. If each source is associated with its own collection of features, the model in effect assesses the significance of the association of a particular group of objects with a collection of features.
- Under the model, the combination of sources that are most strongly associated with a putative cluster can be identified very efficiently.
- In applications for which multiple clusterings of the data have been generated, the model can be used to decide to which clustering a particular candidate cluster is best aligned. This can potentially serve as a foundation upon which multiple clustering criteria can be designed.

References

1. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: Proc. ICDT. (1999)
2. Bennett, K.P., Fayyad, U., Geiger, D.: Density-based indexing for approximate nearest-neighbor queries. In: Proc. KDD. (1999)
3. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM TKDD **3**(1) (2009) 1–58
4. Houle, M.E., Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: Can shared-neighbor-distances defeat the curse of dimensionality? In: Proc. SSDBM. (2010)
5. Bernecker, T., Houle, M.E., Kriegel, H.P., Kröger, P., Renz, M., Schubert, E., Zimek, A.: Quality of similarity rankings in time series. In: Proc. SSTD. (2011)
6. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: Symp. Foundations of Computer Science. (2006) 459–468
7. Houle, M.E., Sakama, J.: Fast approximate similarity search in extremely high-dimensional data sets. In: Proc. ICDE. (2005)
8. Jarvis, R.A., Patrick, E.A.: Clustering using a similarity measure based on shared near neighbors. IEEE TC **C-22**(11) (1973) 1025–1034
9. Guha, S., Rastogi, R., Shim, K.: ROCK: a robust clustering algorithm for categorical attributes,. Inform. Sys. **25** (2000) 345–366
10. Ertöz, L., Steinbach, M., Kumar, V.: Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: Proc. SDM. (2003)
11. Houle, M.E.: The relevant-set correlation model for data clustering. Stat. Anal. Data Min. **1**(3) (2008) 157–176

Cartification: Turning Similarities into Itemset Frequencies*

Bart Goethals

University of Antwerp, Belgium
bart.goethals@ua.ac.be

Extended Abstract

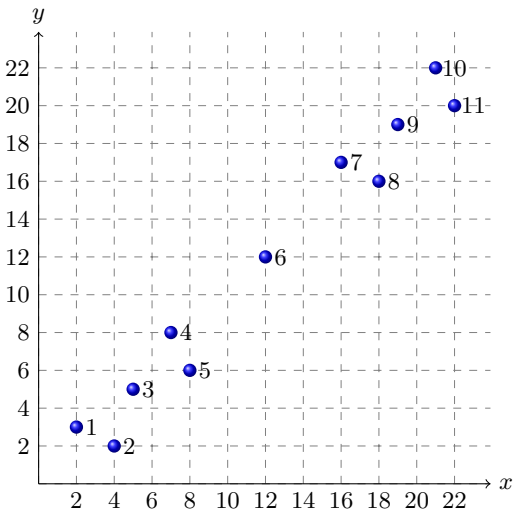
Suppose we are given a multi-dimensional dataset. For every point in the dataset, we create a transaction, or cart, in which we store the k-nearest neighbors of that point for one of the given dimensions. This is repeated for every dimension. The resulting collection of carts can then be used to mine frequent itemsets; that is, sets of points, or clusters, that are frequently seen together in one or more of the dimensions. Essentially, this transformation, that we call *cartification*, combines multiple distance measures without suffering from the curse of dimensionality.

An important observation to make in order to see the potential of cartified data is, that when the frequency of a single item is high, we know it is often found in the neighborhoods of many points in one or more of the dimensions; in fact, we can say that this item lies central in a cluster of data points, and, if it is most central, its frequency will be among the highest of all items in that cluster. Moreover, if an item is indeed part of a cluster, it is easy to see that it will mainly receive its support from those transactions in the database that correspond to the *relevant* dimensions of the cluster, as for the other dimensions it will have wildly varying neighborhoods. This is very important, as it allows us to identify which dimensions are relevant for the cluster, as well as to circumvent the dreaded curse of dimensionality. This observation also goes for sets of items: those itemsets that have relatively high frequency lie centrally in a sub-structure of the data, and will mainly receive their support from the dimensions relevant to that sub-structure. In fact, the latter effect will be even more pronounced for (large) sets than for single items, as it is increasingly unlikely that all points in the itemset lie closely together in an irrelevant dimension.

For example, assume we are given the two-dimensional dataset as shown in Figure 1. For cartification, we first need to choose a threshold k, the number of the nearest neighbors that can be added to each cart. Table 1b shows the resulting database for $k = 3$. The first two columns show the cartification for the x-dimension, and the second two columns for the y-dimension. Every row corresponds to the cart generated from one of the data points. For example, for point 3, the three nearest neighbors in the x-dimension are points $2, 3$, and 4, while for the y-dimension these are $1, 3$, and 5.

In Figure 2a, we plot the frequencies of all items (points) in this cartified database. This plot shows there are two points, 3 and 9, with a high frequency.

* work in collaboration with Jilles Vreeken

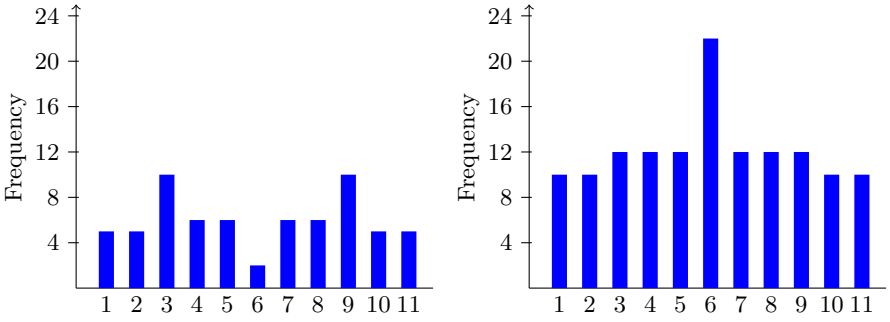


(a) original

x	**cart**	y	**cart**
1	$\{1,2,3\}$	1	$\{1,2,3\}$
2	$\{1,2,3\}$	2	$\{1,2,3\}$
3	$\{2,3,4\}$	3	$\{1,3,5\}$
4	$\{3,4,5\}$	4	$\{3,4,5\}$
5	$\{3,4,5\}$	5	$\{3,4,5\}$
6	$\{5,6,7\}$	6	$\{4,6,8\}$
7	$\{7,8,9\}$	7	$\{7,8,9\}$
8	$\{7,8,9\}$	8	$\{7,8,9\}$
9	$\{8,9,10\}$	9	$\{7,9,11\}$
10	$\{9,10,11\}$	10	$\{9,10,11\}$
11	$\{9,10,11\}$	11	$\{9,10,11\}$

(b) cartified

Fig. 1. Example dataset



(a) k=3

(b) k=6

Fig. 2. Item Frequencies

As Figure 1 shows, these points are in the centers of the clusters $\{1, 2, 3, 4, 5\}$ and $\{7, 8, 9, 10, 11\}$ respectively. Additionally, point 6 has a very low frequency which corresponds to the point being an outlier w.r.t. all other points in the figure.

Obviously, the choice of k is important with regard to which clusters, centers or outliers can be identified; if only, as for k only itemsets of length k or smaller can receive support. Essentially, k affects the granularity at which we are considering centers and outliers. For example, if we look back at the data in Figure 1, as cartified in Table 1b using $k = 3$, we already identified items 3 and 9 as cluster centers, and item 6 as an outlier. If we choose $k = 6$ instead, the resulting frequencies per item are shown in Figure 2b.

Now item 6 has the maximum frequency of 22, far more than all other items, and represents the center of the cluster consisting of all items. At this granularity, no outspoken outliers can be identified, yet the most remote elements (i.e., $1, 2, 10$, and 11) are still identifiable as they have the least support. Hence, increasing k is like zooming out on the data, and so taking a more global point of view.

Let us illustrate this in another example, where we show that not only the centroid items, but also the clusters themselves can be clearly identified. When keeping a relatively close zoom, with $k = 3$, for instance, the largest itemsets with a frequency greater than 2, are $\{1, 2, 3\}, \{3, 4, 5\}, \{7, 8, 9\}$, and $\{9, 10, 11\}$. These sets correspond to the clusters in the data when we aim at finding small clusters. Next, we zoom out to $k = 5$, and now find $\{1, 2, 3, 4, 5\}$ and $\{7, 8, 9, 10, 11\}$ as the largest itemsets with support greater than 2. These sets represent the clusters in the data well. Note however, these are good clusters *at the current level of zoom*. Indeed, if we zoom in 'too much', every point is regarded as a cluster, and if we zoom out 'too much' all of the data automatically becomes one big cluster. Clearly, what is 'enough' for the task at hand is subjective, and as clustering is explorative in nature, it is ultimately up to the data analyst to decide what types of clusters are potentially of interest, and hence, how to set k.

To conclude, preliminary cartification experiments on real data show that it allows us to efficiently discover centroid and outlying *sets of points*, subspace clusters, and clusterings, while not suffering from the curse of dimensionality. Multiple dimensions, numerical or categorical, as well as multiple distance measures, can be combined, and become represented in the frequency of clusters. Moreover, several efficient and scalable itemset mining techniques can be effectively applied on the cartified database, resulting in meaningful and interesting discoveries.

When Pattern Met Subspace Cluster
a Relationship Story

Jilles Vreeken[1] and Arthur Zimek[2]

[1] Advanced Database Research and Modeling, Universiteit Antwerpen, Belgium
http://www.adrem.ua.ac.be/~jvreeken
jilles.vreeken@ua.ac.be

[2] Ludwig-Maximilians-Universität München, Germany
http://www.dbs.ifi.lmu.de/~zimek
zimek@dbs.ifi.lmu.de

Abstract. While subspace clustering emerged as an application of pattern mining and some of its early advances have probably been inspired by developments in pattern mining, over the years both fields progressed rather independently. In this paper, we identify a number of recent developments in pattern mining that are likely to be applicable to alleviate or solve current problems in subspace clustering and vice versa.

1 Introduction

Arguably, the two main proponents of exploratory data mining research are clustering and pattern mining. Both share the aim of the field—extracting interesting, non-trivial, and previously unknown knowledge from data—yet, they are orthogonal in their approach, or at least appear so at first glance.

Pattern mining, to start with, is concerned with developing theory and algorithms for identifying groups of attributes and some selection criteria on those; such that the most 'interesting' attribute-values combinations in the data are returned. That is, by that selection we identify objects in the data that somehow stand out. The prototypical example is supermarket basket analysis, in which by frequent itemset mining we identify items that are frequently sold together–with the infamous *beer* and *nappies* as an example of an interesting pattern.

Clustering, on the other hand, in general aims at finding groups of similar objects in a database. Aside from algorithmic variations in the process of identifying these groups, the major differences between various clustering approaches is in the actual meaning of 'similar'. Especially in high dimensional data the notion of similarity is not a trivial one. The so-called 'curse of dimensionality' is often given as the main motivation for 'subspace clustering' [34], where our goal is to identify both sets of objects, as well as subsets of attributes (subspaces), on which those objects are measured to be highly similar. As such, we see that both pattern mining and subspace clustering identify sub-parts of the data.

In this paper we explore and discuss a number of connections between these two active fields of research. We argue that research in subspace clustering,

having a common origin with pattern mining and sharing some early ideas, has deviated from the route of pattern mining subsequently. Interestingly, both fields now face problems already studied by the other. Here, we would like to point out interesting recent research topics on pattern mining where research on subspace clustering can possibly benefit from, and vice versa. For example, the explosion in numbers of results, and reducing their redundancy, are currently open problems in subspace clustering but have recently been studied in detail by the pattern mining community. On the other hand, the notion of alternative results, as well as the generalization beyond binary data, are topics where pattern miners may draw much inspiration from recent work in (subspace) clustering.

The goal of this paper is to identify a number of developments in these fields that should not go unnoticed; we are convinced that solutions for pattern mining problems are applicable in subspace clustering, and vice versa. In other words, it is time to meet the relatives.

The remainder of this paper is organized as follows. First, we discuss the background of subspace clustering, and how it relates to pattern mining. Next, we go into the similarities between their results. Section 4 discusses developments in pattern mining that are interesting with regard to subspace clustering—and vice versa in Section 5. We round up with conclusions in Section 6.

2 It's a Family Affair

2.1 The Curse

The so-called 'curse of dimensionality' is often credited for causing problems in similarity computations in high dimensional data, and, hence, is given as motivation for specialized approaches such as 'subspace clustering' [34]. Let us consider two aspects of the 'curse' that are often confused in the literature: (i) the concentration effect of L_p-norms and (ii) the presence of irrelevant attributes.

Regarding the concentration effect (i), the key result of [10] states that, if the ratio of the variance of the length of any point vector $\boldsymbol{x} \in \mathbb{R}^d$ (denoted by $\|\boldsymbol{x}\|$) with the length of the mean point vector (denoted by $E[\|\boldsymbol{x}\|]$) converges to zero with increasing data dimensionality, then the proportional difference between the farthest-point distance D_{max} and the closest-point distance D_{min} (the *relative contrast*) vanishes, i.e., all distances concentrate around a mean, and look alike. This observation is often quoted for motivating subspace clustering as a specialized procedure. It should be noted, though, that the problem is neither well enough understood (see e.g. [20]) nor actually relevant when the data follows different, well separated distributions [8, 9, 29].

Regarding the separation of clusters, the second problem (ii) is far more important for subspace clustering: In order to find structures describing phenomena, abundances of highly detailed data are collected. Among the features of a high dimensional data set, for any given query object, many attributes can be expected to be irrelevant to describing that object. Irrelevant attributes can easily obscure clusters that are clearly visible when we consider only the relevant

'subspace' of the dataset. Hence, they interfere with the performance of similarity measures in general, but in a far more fundamental way for clustering. The relevance of certain attributes may differ for different groups of objects within the same data set. Thus, many subsets of objects are defined only by some of the available attributes, and the irrelevant attributes ('noise') will interfere with the efforts to find these subsets. This second problem is actually the true motivation for designing specialized methods to look for clusters in subspaces of a dataset.

2.2 Variants

In general, in subspace clustering similarity is defined in some relation to subsets or combinations of attributes or dimensions of database objects. Hence, a clustering with n clusters for a database $\mathcal{D} \times \mathcal{A}$, with the set of objects $\mathcal{D}$ and with the full set of attributes $\mathcal{A}$, can be seen as a set $\mathcal{C} = \{(\mathcal{C}_1, \mathcal{A}_1), \ldots, (\mathcal{C}_n, \mathcal{A}_n)\}$, where $\mathcal{C}_i \subseteq \mathcal{D}$ and $\mathcal{A}_i \subseteq \mathcal{A}$, i.e., a cluster is defined w.r.t. a set of objects and w.r.t. a set of attributes (i.e., a subspace).

Subspace clustering algorithms are typically split into two groups; in 'projected clustering' objects belong to at most one cluster, while 'subspace clustering' (in a more narrow sense) seeks to find all possible clusters in all available subspaces, allowing overlap [34]. The distinction (and terminology) originates from the two pioneering papers in the field, namely CLIQUE [2] for 'subspace clustering' and PROCLUS [1] for projected clustering; and the two definitions allow a broad field of hybrids. Since we are interested in the relationship between pattern mining and subspace clustering, we will let aside projected clustering and hybrid approaches and concentrate on subspace clustering in the narrower sense as defined in [2]. In this setting, subspace clustering is usually related to a bottom-up traversal of the search space of all possible subspaces, i.e., starting with all one dimensional subspaces, two-dimensional combinations of these subspaces, three dimensional combinations of the two dimensional subspaces and so on, all (relevant) subspaces are searched for clusters residing therein.

Considering CLIQUE, we find the intuition of subspace clustering promoted there closely related to pattern mining. To this end, we consider frequent itemset mining [3], in which we consider binary transaction data, where transactions are sets of items A, B, C, etc. The key idea of APRIORI [3] is to start with itemsets of size 1 that are frequent, and exclude all itemsets from the search that cannot be frequent anymore, given the knowledge which smaller itemsets are frequent. For example, if we count a 1-itemset containing A less than the given minimum support threshold, all 2-itemsets, 3-itemsets, etc. containing A (e.g., $\{A, B\}, \{A, C\}, \{A, B, C\}$) cannot be frequent either and need not be considered. While theoretically the search space remains exponential, in practice searching becomes feasible even for very large datasets.

Transferring this problem to subspace clustering, each attribute represents an item, and each subspace cluster is then an itemset containing the items representing the attributes of the subspace. This way, finding itemsets with support 1 relates to finding all combinations of attributes constituting a subspace of at least

one cluster. This observation is the rationale of most bottom-up subspace clustering approaches: subspaces containing clusters are determined starting from all 1-dimensional subspaces accommodating at least one cluster, employing a search strategy similar to that of itemset mining algorithms. To apply any efficient algorithm, the cluster criterion must implement a downward closure property (i.e. (anti-)monotonicity): *If subspace $\mathcal{A}_i$ contains a cluster, then any subspace $\mathcal{A}_j \subseteq \mathcal{A}_i$ must also contain a cluster.* The anti-monotonic reverse implication, *if a subspace $\mathcal{A}_j$ does not contain a cluster, then any superspace $\mathcal{A}_i \supseteq \mathcal{A}_j$ also cannot contain a cluster*, can subsequently be used for pruning.

Clearly, this is a rather naïve use of the concept of frequent itemsets in subspace clustering. What constitutes a good subspace clustering result is defined here apparently in close relationship to the design of the algorithm, i.e., the desired result appears to be defined according to the expected result (as opposed to: in accordance to what makes sense) — see the discussion in [34]. Resulting clusters are usually highly redundant and, hence, mostly useless.

This issue is strongly related to the so-called pattern explosion. Taking frequent itemset mining as an example, we see that for high minimal support thresholds, only trivial results are returned, but that for lower thresholds we end up with enormous amounts of results—a collection that is highly redundant, and many of the returned patterns are variations of the same theme. Recently, pattern miners have started to acknowledge they have been asking the wrong question: instead of asking for *all* patterns that satisfy some constraints, we should ask for small, non-redundant, and high quality *sets of patterns*—where by high-quality we mean that each of the patterns in the set satisfy the thresholds we set on interestingness or similarity, and the set is optimal with regard to some criterion, e.g. mutual information [32], compression [49], area [21].

Research on subspace clustering inherited all the deficiencies from this originally ill-posed problem. However, early research on subspace clustering as follow-ups of CLIQUE apparently also tried to transfer improvements from pattern mining. As an example, ENCLUS [14] uses several quality criteria for subspaces, not only implementing the downward closure property, but also an upward closure (i.e., allowing search for interesting subspaces as specializations as well as generalizations). This most probably relates to the concept of positive and negative borders known from closed frequent itemsets [46]. Both can be seen as implementations of the classical concept of version spaces [38].

3 I Say Pattern, You Say Subspace Cluster

Methods aside, there are two notions we have to discuss that do, or do not, make the two fields different. First and foremost, what is a result? And, second, can we relate interestingness and similarity? To start with the former, in subspace clustering, a single result defined by the Cartesian product of objects $C \subseteq \mathcal{D}$ and attributes $A \subseteq \mathcal{A}$. In order to be considered as a result, each of the objects in the selection should be similar to the others, over the selected attributes, according to the employed similarity function. In order to make a natural connection to

pattern mining, we adopt a visual approach; if we are allowed to re-order both attributes and objects freely, we can reorder $\mathcal{D}$ and $\mathcal{A}$ such that C and A define a rectangle in the data, or a *tile*. In pattern mining, the notion of a tile has become very important in recent years [17, 21, 23, 33]. Originally the definition of a pattern was very much along the lines of an SQL query, posing selection criteria on which objects in the data are considered to *support* the pattern or not. As such, beyond whether they contribute to such a global statistic, the selected objects were not really taken into account. In many recent papers, however, the supporting objects are explicitly taken into account, and by doing so, patterns also naturally define *tiles*. In the next section we will link this approach to the reduction of redundancy. So, both fields identify *tiles*, sub-parts of the data. However, both have different ways of arriving at these tiles. In pattern mining, results are typically selected by some measure of 'interestingness'—of which support, the number of selected objects, is the most well-known example. In subspace clustering, on the other hand, we measure results by how similar the selected objects are over the considered attributes. Clearly, while this may lead to discovering rather different tiles, it is important to realize that both approaches do find tiles, and provide some statistics for the contents of each tile.

We observe that in pattern mining the selection of the objects 'belonging' to the pattern is very strict—and that as such those objects will exhibit high similarity over the subspace of attributes the pattern identifies. For example, in standard frequent itemset mining, transactions (i.e., objects) are only selected if they are a strict superset of the pattern at hand—and in fault-tolerant itemset mining typically only very few attribute-value combinations are allowed to deviate from the template the pattern identifies. Linking this to similarity, in this strict selection setting, it is easy to see that for the attributes identified by the pattern, the selected objects are highly similar. The same also holds for subgroup discovery, a supervised branch of pattern mining. In subgroup discovery the patterns typically strongly resemble SQL range-based selection queries, where the goal is to identify those patterns (intention) that select objects (extension) that correlate strongly to some target attribute(s). Intuitively, the more strict the selection criteria are per attribute, the more alike the selected objects will be on those attributes. So, in the traditional sense, patterns identified as interesting by a measure using support, are highly likely to correspond to highly-similar subspace clusters, the more strict conditions the pattern defines on its attributes. The other way around, we can say that the higher the similarity of a subspace cluster, the easier it will be to define a pattern that covers the same area of the database. And, the larger this highly-similar subspace cluster is, the more likely it is that it will be discovered by pattern mining using any support-based interestingness measure.

Besides this link, it is interesting to consider what the main differences are. In our view, it is a matter of perspective; whether to take a truly local stance at the objects, and from *within* a tile, like in subspace clustering, or, whether to take a slightly more global stance and look at how we can select those objects by defining conditions on the attributes. Further, we remark that both interest-

ingness and similarity are very vague concepts, for which many proposals exist. A unifying theory, likely from a statistical point of view, would be very welcome.

4 Advances of Interest in Pattern Mining

In this section we discuss some recent advances in pattern mining research that may likewise be applicable for issues in subspace clustering.

4.1 Summarizing Sets of Patterns

As touched upon in Section 2, reducing redundancy has been studied for a long period of time in pattern mining. Very roughly speaking, two main approaches can be distinguished: summarizing the result set, and summarizing the data.

In this subsection we discuss the former, in which we find well-known examples. The main idea of this approach is that we have a set of results $\mathcal{F}$, consisting of results that have passed the constraints that we have set, e.g. they all pass the interestingness threshold. Now, with the goal of reducing redundancy in $\mathcal{F}$, we want to select a subset $\mathcal{S} \subseteq \mathcal{F}$ such that $\mathcal{S}$ contains as much information on the whole of $\mathcal{F}$ while being minimal in size.

Perhaps the most well-known examples of this approach are *closed* [46] and *maximal* [7] frequent itemsets, by which we only allow elements $X \in \mathcal{F}$ into $\mathcal{S}$ for which no superset exists that has the same support, or no superset exists that does not meet the mining constraints, respectively. As such, for closed sets, given $\mathcal{S}$ we can reconstruct $\mathcal{F}$ without loss of information—and for maximal sets we can reconstruct only the itemsets, not their frequencies. Non-derivable itemsets [13] follow a slightly different approach, and only provide those itemsets for which their frequency cannot be derived from the rest. While the concepts of closed and maximal have been applied in subspace clustering, non-derivability has not been explored yet, to the best of our knowledge.

Reduction by closure only works well when data are highly structured, and it deteriorates rapidly with noise. A recent improvement is margin-closedness [39], where we consider elements into the closure for which our measurement falls within a given margin. This provides strong reduction in redundancy, and higher noise resistance; we expect it to be applicable for subspace clusters.

Perhaps not trivially translatable to subspace clusters, another branch of summarization is that of picking or creating a number of representative results. Yan et al. [51] choose $\mathcal{S}$ such that the error of predicting the frequencies in $\mathcal{F}$ is minimized. Here, it may well be reasonable to replace frequency with similarity. There are some attempts in this direction, e.g. in biclustering [48].

More examples exist, but for reasons of space, we continue to a more important development of reducing redundancy.

4.2 Pattern Set Mining

While the above-mentioned techniques do reduce redundancy, they typically still result in large collections of patterns, that still do contain many variations of the

same theme. As stated in Section 2, a recent major insight in pattern mining is that we were asking the wrong question. Instead of asking for all patterns that satisfy some constraints, we should be asking for a small non-redundant group of high-quality patterns. With this insight, the attention shifted from attempting to summarize the full result $\mathcal{F}$, to provide a good summarization of the *data*. In terms of subspace clustering, this means that we would select that group of subspace clusters such that we can approximate (explain, describe, etc.) the data optimally. Here we discuss a few examples of such *pattern set mining* techniques that we think are applicable to subspace clustering in varying degrees.

The most straightforward technique we discuss is tiling [21]. It proposes to not just consider itemsets, but also the transactions they occur in—the same notion we adopted in Section 3. The main idea here is that patterns that cover a large area are more interesting than patterns of a small area, where area is defined as the product of the number of items and number of transactions that support the itemset. Most importantly, the authors give an algorithm for approximating the optimal *tiling* of the data—those k tiles that together cover as much of the data as possible. As the paper only considers exact tiles, for which exactly what the data values are, namely 1s, the returned tilings are good summarizations of the data. It is not trivially translated to subspace clustering. One could extract a cluster profile, e.g. a centroid, and take the deviation between the current summary and the real data into account—something that one could minimize.

In this direction, other promising approaches take cues from Information Theory, the Minimum Description Length principle [25] in particular. That is, they approach the pattern set selection problem from the perspective of lossless compression; the best set of patterns is that set of patterns that together compress the data best. Gionis et al. [23] propose a hierarchical model for discovering informative regions (patterns, subspaces) in the data by employing MDL. It does not consider a candidate set $\mathcal{F}$, but looks for interesting regions directly, assuming a given fixed order of the attributes and objects. The hierarchical nature potentially does link strongly to subspace clustering, where we could consider nested clusters—a related method for clusters was proposed by Böhm et al [12]. Siebes et al. [49] proposed the Krimp algorithm to approximate the set of itemsets that together optimally compress the data from a candidate collection $\mathcal{F}$. The resulting code tables have been shown to be of very high quality, while reducing the number of patterns up to 7 orders of magnitude [49]. In turn, Kontonasios and De Bie [33] combine the ideas of MDL and Tiling, although they do not simply accumulate tiles to maximize the covered area of the database. Instead, they measure how informative candidate results are with regard to a static probabilistic background model, while also taking their complexity into account. In other words, how many bits does adding result X save us when explaining the data, and how many does it cost to understand X.

Each of the above methods have, as of yet, only been defined for (single and multi-table) binary data. However, MDL theory does exist for richer data types, and we would like to point out the strong potential for reducing redundancy in subspace clustering by aiming at that set of subspace clusters that together

describe the data best. That is, those clusters by which we can encode the data and the model most succinctly. Note that this approach naturally allows for overlapping clusters, as well as refinements (a big general cluster, and a smaller sub-region of it)—results will be selected if they provide sufficient extra information by which the data can be compressed better than without, while not costing too much to be described themselves.

4.3 Significance and Randomization

Perhaps the largest problem in exploratory data mining is validation. Unlike in settings where there is a clear formal goal, such as in many supervised machine learning, our goal is as ill-defined as to find 'interesting things'. Like in clustering a plethora of different similarity measures have been considered, all of which may identify some interesting interplay between objects, also in pattern mining a broad spectrum of interestingness measures have been discussed, yet there is no gold standard by which we can compare results.

One approach that recently received ample attention in pattern mining is that of statistical significance. If a result can be easily explained by background knowledge, it will most likely not be interesting to the end user, never mind how large its support or similarity. Webb [50] proposes to rank patterns depending on their individual statistical significance. A more general framework was proposed by Gionis et al. [22], who propose to investigate significance of results *in general* through randomization. To this end, they introduce swap randomization as a means to sample random binary datasets of the same row and column margins as the original data, and so calculate empirical p-values. Ojala et al. [44,45] gave variants for numerical data, easing the use of the model for subspace clustering. Hanhijarvi et al. [28] extended the framework such that more complex background information, such as cluster densities and itemset frequencies, can be entered into the model—making the approach applicable for iterative data mining. De Bie [17] proposed to model these probability distributions over datasets *analytically*, by employing the Maximum Entropy principle. A main advantage is that this allows for the calculation of exact p-values.

As of yet, with the exception of the latter, each of the above have already been formalized for a wide variety of data types, and, hence, we expect these methods to be rather easily applicable for assessing whether a subspace cluster, subspace clustering or multiple clustering is significant—whether in light of some basic properties of the data, or with regard to more involved known structure.

5 Interesting Advances in Subspace Clustering

In this section we discuss advances in subspace clustering that may be of particular worth in progressing pattern mining research.

5.1 Half of Success is Knowing When to Stop

In early approaches to subspace clustering, the fixed grid, that allows an easy translation to frequent itemsets, introduces bias towards certain cluster properties. Thus, it has found major interest in research on subspace clustering. The MAFIA [43] method uses an adaptive grid, while its generation of subspace clusters is similar to CLIQUE. Another variant, nCluster [37], allows overlapping windows of length δ as 1-dimensional units of the grid. SUBCLU [31] uses the DBSCAN cluster model of density-connected sets [18], letting go the grid-approach completely. Nevertheless, density-connected sets satisfy the downward closure property. This enables SUBCLU to search for density-based clusters in subspaces also in an APRIORI-like style. A global density threshold, as used by SUBCLU and the grid-based approaches, leads to a bias towards a certain dimensionality: a tighter threshold separates clusters from noise well in low dimensions, but tends to loose clusters in higher dimensions. A more relaxed threshold detects high dimensional clusters but produces an excessive amount of low dimensional clusters. This problem has been of major interest in the research on subspace clustering in the recent years. See e.g. [4, 42], where the density-threshold is adapted in turn to the dimensionality of the subspace currently being scrutinized during the run of the algorithm.

A problem closely related to choosing the appropriate density level is the redundancy issue, that also found much interest recently [5, 26, 41]. These approaches aim at reporting only the most representative of a couple of redundant subspace clusters. While technically the approaches differ, in concept, adaptive density-thresholds show high similarity with selection of patterns based on statistical significance [33, 50]. Significance of subspace clusters, though, has only be addressed once so far [40]. Hence, we regard it as highly likely that both approaches can learn from each other's endeavours.

5.2 Alternatively...

A recent development in both fields is finding alternatives to results. The techniques we employ in exploratory data mining can only seldom be shown to provide optimal results, instead typically returning good results heuristically. However, while one good result might shed light on one aspect of the data, it might ignore other parts of the data—for which other results will be informative. A clustering result that can be judged valid by known structures may even be completely uninteresting [19]. Hence a proposal to improve cluster evaluation relies on the *deviation* of a clustering from known structures instead of judging the coverage of known structures [35].

This is almost literally the approach taken in the subfield of alternative clustering, where one wants to discover a good clustering that is orthogonal from what we already found, or, alternatively, where we want to find n good clusterings each of which be different from any other. Approaches for finding alternative clusterings mostly use ensemble techniques [6, 15, 16, 24]. A key requirement for building good ensembles is a source of diversity for the ensemble members.

Clearly, using different feature subsets (i.e., subspaces) can be a very good source of diversity and actually has occasionally been used in alternative clustering as one possibility to find different clustering solutions [47]. Alternative clustering approaches typically seek diversity constrained by non-redundancy. Hence, allowing some degree of redundancy could be meaningful, such as allowing overlap between clusters. In different subspaces, one subset of objects could belong to two different, yet meaningful, clusters. While this would increase their redundancy, reporting both of them instead of only one would not necessarily be useless. Considerations in this direction can be found w.r.t. subspace clustering [27].

Also it has been conceded that preserving known properties or concepts is desirable when seeking different clusters [47]. As searching for subspaces that are (at least partially) different from subspaces of already found clusters, the more specialized area of multiview clustering [11, 30] is also of interest here, and can be seen as a special case of seeking alternative results. The constraint here is the orthogonality of subspaces. It can also be seen as a special case of subspace clustering allowing maximal overlap yet seeking minimally redundant clusters by accommodating different concepts (as proposed e.g. in [27]). These approaches highlight that highly overlapping clusters in different subspaces (i.e., certain subsets of objects may belong to several clusters simultaneously) need not be redundant nor meaningless (see also the discussion in [19, 36]).

6 Conclusion

There exist strong links between Subspace Clustering and Pattern Mining, although the topics of research within the two fields have diverged over time. We argued the case that both fields are not as different as they might think, and moreover, that both can learn much from the experience gained by the other. In other words, we say it is time for the two fields to meet again. To this end, we gave a (far from complete) overview of proposals from the one field that we find have strong potential to advance research in the other, and vice versa.

Acknowledgements Jilles Vreeken is supported by a Post-Doctoral Fellowship of the Research Foundation – Flanders (FWO).

References

1. C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proc. SIGMOD*, 1999.
2. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. SIGMOD*, 1998.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. SIGMOD*, 1994.
4. I. Assent, R. Krieger, E. Müller, and T. Seidl. DUSC: dimensionality unbiased subspace clustering. In *Proc. ICDM*, 2007.

5. I. Assent, E. Müller, S. Günnemann, R. Krieger, and T. Seidl. Less is more: Non-redundant subspace clustering. In *Proc. ACM SIGKDD Workshop MultiClust*, 2010.
6. E. Bae and J. Bailey. COALA: a novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In *Proc. ICDM*, 2006.
7. R. Bayardo. Efficiently mining long patterns from databases. In *Proc. SIGMOD*, pages 85–93, 1998.
8. K. P. Bennett, U. Fayyad, and D. Geiger. Density-based indexing for approximate nearest-neighbor queries. In *Proc. KDD*, 1999.
9. T. Bernecker, M. E. Houle, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. Quality of similarity rankings in time series. In *Proc. SSTD*, 2011.
10. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proc. ICDT*, 1999.
11. S. Bickel and T. Scheffer. Multi-view clustering. In *Proc. ICDM*, 2004.
12. C. Böhm, F. Fiedler, A. Oswald, C. Plant, B. Wackersreuther, and P. Wackersreuther. ITCH: information-theoretic cluster hierarchies. In *Proc. ECML PKDD*, 2010.
13. T. Calders and B. Goethals. Non-derivable itemset mining. *Data Min. Knowl. Disc.*, 14(1):171–206, 2007.
14. C. H. Cheng, A. W.-C. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proc. KDD*, pages 84–93, 1999.
15. X. H. Dang and J. Bailey. Generation of alternative clusterings using the CAMI approach. In *Proc. SDM*, 2010.
16. I. Davidson, S. S. Ravi, and L. Shamis. A SAT-based framework for efficient constrained clustering. In *Proc. SDM*, 2010.
17. T. De Bie. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Min. Knowl. Disc.*, 2010.
18. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD*, 1996.
19. I. Färber, S. Günnemann, H.-P. Kriegel, P. Kröger, E. Müller, E. Schubert, T. Seidl, and A. Zimek. On using class-labels in evaluation of clusterings. In *Proc. ACM SIGKDD Workshop MultiClust*, 2010.
20. D. Francois, V. Wertz, and M. Verleysen. The concentration of fractional distances. *IEEE TKDE*, 19(7):873–886, 2007.
21. F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *Proc. DS'04*, pages 278–289, 2004.
22. A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *ACM TKDD*, 1(3), 2007.
23. A. Gionis, H. Mannila, and J. K. Seppänen. Geometric and combinatorial tiles in 0-1 data. In *Proc. ECML PKDD'04*, 2004.
24. D. Gondek and T. Hofmann. Non-redundant data clustering. In *Proc. ICDM*, 2004.
25. P. D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.
26. S. Günnemann, I. Färber, E. Müller, and T. Seidl. ASCLU: alternative subspace clustering. In *Proc. ACM SIGKDD Workshop MultiClust*, 2010.
27. S. Günnemann, E. Müller, I. Färber, and T. Seidl. Detection of orthogonal concepts in subspaces of high dimensional data. In *Proc. CIKM*, 2009.
28. S. Hanhijärvi, M. Ojala, N. Vuokko, K. Puolamäki, N. Tatti, and H. Mannila. Tell me something I don't know: randomization strategies for iterative data mining. In *Proc. KDD*, pages 379–388, 2009.

29. M. E. Houle, H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Can shared-neighbor distances defeat the curse of dimensionality? In *Proc. SSDBM*, 2010.
30. P. Jain, R. Meka, and I. S. Dhillon. Simultaneous unsupervised learning of disparate clusterings. *Stat. Anal. Data Min.*, 1(3):195–210, 2008.
31. K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *Proc. SDM*, 2004.
32. A. J. Knobbe and E. K. Y. Ho. Pattern teams. In *Proc. PKDD*, 2006.
33. K.-N. Kontonasios and T. DeBie. An information-theoretic approach to finding informative noisy tiles in binary databases. In *Proc. SDM*, 2010.
34. H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM TKDD*, 3(1):1–58, 2009.
35. H.-P. Kriegel, E. Schubert, and A. Zimek. Evaluation of multiple clustering solutions. In *Proc. ECML PKDD Workshop MultiClust*, 2011.
36. H.-P. Kriegel and A. Zimek. Subspace clustering, ensemble clustering, alternative clustering, multiview clustering: What can we learn from each other? In *Proc. ACM SIGKDD Workshop MultiClust*, 2010.
37. G. Liu, J. Li, K. Sim, and L. Wong. Distance based subspace clustering with flexible dimension partitioning. In *Proc. ICDE*, 2007.
38. T. M. Mitchell. Version spaces: A candidate elimination approach to rule learning. In *Proc. IJCAI*, 1977.
39. F. Moerchen, M. Thies, and A. Ultsch. Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression. *KAIS*, 2010.
40. G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *Proc. KDD*, 2008.
41. E. Müller, I. Assent, S. Günnemann, R. Krieger, and T. Seidl. Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In *Proc. ICDM*, 2009.
42. E. Müller, I. Assent, R. Krieger, S. Günnemann, and T. Seidl. DensEst: density estimation for data mining in high dimensional spaces. In *Proc. SDM*, 2009.
43. H.S. Nagesh, S. Goil, and A. Choudhary. Adaptive grids for clustering massive data sets. In *Proc. SDM*, 2001.
44. M. Ojala. Assessing data mining results on matrices with randomization. In *Proc. ICDM*, 2010.
45. M. Ojala, N. Vuokko, A. Kallio, N. Haiminen, and H. Mannila. Randomization methods for assessing data analysis results on real-valued matrices. *Stat. Anal. Data Min.*, 2(4):209–230, 2009.
46. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. ICDT*, 1999.
47. Z. J. Qi and I. Davidson. A principled and flexible framework for finding alternative clusterings. In *Proc. KDD*, 2009.
48. E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(Suppl. 1):S243–S252, 2001.
49. J. Vreeken, M. van Leeuwen, and A. Siebes. Krimp: Mining itemsets that compress. *Data Min. Knowl. Disc.*, 23(1):169–214, 2010.
50. G. I. Webb. Discovering significant patterns. *Mach. Learn.*, 68(1):1–33, 2007.
51. X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: a profile-based approach. In *Proc. KDD*, 2005.

Fast Multidimensional Clustering of Categorical Data

Tengfei Liu[1], Nevin L. Zhang[1], Kin Man Poon[1], Yi Wang[2], and Hua Liu[1]

[1] Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{liutf, lzhang, lkmpoon, aprillh}@cse.ust.hk

[2] Department of Computer Science
National University of Singapore
wangy@comp.nus.edu.sg

Abstract. Early research work on clustering usually assumed that there was one true clustering of data. However, complex data are typically multifaceted and can be meaningfully clustered in many different ways. There is a growing interest in methods that produce multiple partitions of data. One such method is based on latent tree models (LTMs). This method has a number of advantages over alternative methods, but is computationally inefficient. We propose a fast algorithm for learning LTMs and show that the algorithm can produce rich and meaningful clustering results in moderately large data sets.

Keywords: Multidimensional Clustering, Model-based Clustering, Latent Tree Models

1 Introduction

There are several clustering methods that produce multiple partitions. We refer to them as *multi-partition clustering (MPC)* methods. MPC methods, according to the way that partitions are found, can be divided into two categories: *sequential MPC* methods and *simultaneous MPC* methods.

Sequential MPC methods produce multiple clustering solutions sequentially. One such kind of method is known as *alternative clustering* [1–3]. It aims to discover a new clustering that is different from a previously known clustering. The key issue is how to ensure the novelty of the new clustering. One can repeatedly apply such methods to produce a sequence of clusterings.

Simultaneous MPC methods, on the other hand, produce multiple clustering solutions simultaneously. Both distance-based and model-based methods have been proposed for simultaneous MPC. The distance-based methods proposed by [4, 5] require as inputs the number of partitions and the number of clusters in each partition. They try to optimize the quality of each individual partition while keeping different partitions as dissimilar as possible. Model-based methods fit data with a probabilistic model that contains multiple latent variables. Each latent variable represents a soft partition and can be viewed as a hidden dimension of data. So we refer to model-based simultaneous MPC also

as *multidimensional clustering (MDC)*. Unlike distance-based methods, model-based methods can automatically determine the numbers of partitions and the number of clusters in each partition based on statistical principles.

Among the MDC methods, Galimberti et al. [6] and Guan et al. [7] use what we call *disjoint and independent view (DIV) models*. In a DIV model, each latent variable is associated with a subset of attributes, which gives one *view* of data. The subsets for different latent variables are disjoint. A latent variable is independent of all the other latent variables and all the attributes that are not associated with it. On the other hand, Zhang [8] and Poon et al. [9] use latent tree models (LTMs). An LTM can be viewed as a DIV model with the latent variables connected to form a tree structure.

This paper is concerned with the use of LTMs for producing multiple clustering solutions. Currently, there is a lack of efficient algorithms for learning LTMs. The fastest algorithm takes weeks to process data sets with around 100 attributes and a few thousands samples. We propose a more efficient algorithm that can analyze data with hundreds of attributes in a few hours. We also provide empirical results to show that the algorithm can produce much richer clustering results than alternative methods.

2 Latent Tree Models

Technically an LTM is a Markov random field over an undirected graph, where variables at leaf nodes are observed and variables at internal nodes are hidden. An example of LTM is shown in Figure 1. It is part of the latent tree model learned from WebKB data which will be described in more details in Section 4.2. The Y-variables are the latent variables. The numbers in parenthesis are the numbers of states of the latent variables. The leaf nodes here are different words which take binary values to indicates presence or absence of the word. The edges represent probabilistic dependence and their width represents dependence strength.

For technical convenience, we often root an LTM at one of its latent nodes and regard it as a directed graphical model, i.e., a Bayesian network. For the model in Figure 1, suppose we use Y_{43} as the root. Then the edges are directed as pointing away from Y_{43}. For example, the edges $Y_{43} - Y_{53}$ and $Y_{53} -$ ut should be directed as $Y_{43} \rightarrow Y_{53}$ and $Y_{53} \rightarrow$ ut. The numerical information of the model includes a marginal distribution $P(Y_{43})$ for the root and one conditional distribution for each edge. For the edge $Y_{43} \rightarrow Y_{53}$, we have distribution $P(Y_{53}|Y_{43})$; for the edge $Y_{53} \rightarrow$ ut, we have distribution $P(\mathsf{ut}|Y_{53})$; and so on. The product of those distributions defines a joint distribution over all the latent and observed variables. In this paper, we assume all variables are discrete.

2.1 The State of the Art

Suppose there is a data set $\mathbf{D}$. The attributes of the data set are the observed variables. To learn an LTM from $\mathbf{D}$, one needs to determine: (1) the number of latent variables, (2) the number of states of each latent variable, (3) the connections among the latent and observed variables, and (4) the probability

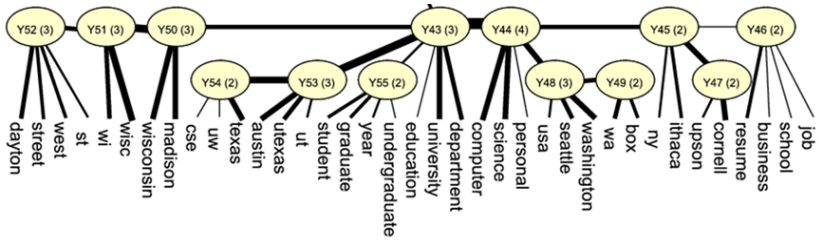


Fig. 1. Part of the latent tree model produced by new algorithm on the WebKB data.

parameters. We use m to denote the information for the first three items and θ to denote the collection of parameter values. We aim at finding the pair (m, θ^*) where θ^* is the maximum likelihood estimate of the parameters and m maximizes the BIC score [10]:

$$BIC(m \mid \mathbf{D}) = \log P(\mathbf{D}|m, \theta^*) - \frac{d(m)}{2} \log N$$

where $d(m)$ is the number of free parameters in m and N is the sample size.

Several algorithms for learning LTMs have been proposed. The state-of-the-art is an algorithm called EAST [11]. It is a search-based method and is capable of producing good models for data sets with dozens of attributes. However, it is not efficient enough for data sets with more than 100 attributes. There are two other algorithms that are more efficient than EAST, but they focus on special LTMs. Harmeling and Williams [12] consider only binary trees, while Choi et al. [13] assume all the variables share the same domain. Neither method is intended for cluster analysis.

3 The Bridged Islands Algorithm

We now set out to present a new algorithm that is drastically more efficient than EAST. In an LTM, the set of attributes that are connected to the same latent variable is called a *sibling cluster*. Attributes in the cluster are said to be *siblings*. In the LTM shown in Figure 1, attributes austin, utexas and ut form one sibling cluster because they are all connected to Y_{53}.

The new algorithm proceeds in five steps:

1. Partition the set of attributes into sibling clusters;
2. For each sibling cluster introduce a latent variable and determine the number of states of this variable;
3. Determine the connections among the latent variables so that they form a tree;
4. Determine the values of the probability parameters;
5. Refine the model.

If we imagine the sibling clusters formed in Step 1, together the latent variables added in Step 2, as islands in an ocean, then the islands are connected in Step 3. So we call the algorithm the *bridged islands (BI)* algorithm. In the following, we describe each step of BI in details.

Step 1: BI determines sibling clusters using two intuitions. First, attributes from the same sibling cluster tend to be more closely correlated than those from

different sibling clusters. Second, if two attributes are siblings in the optimal model for one set of attributes, they should also be siblings in the optimal model for a subset of the attributes.

BI determines the first sibling cluster as follows. There is a working subset of attributes. Initially, it contains the pair of attributes with the highest mutual information(MI). Here MI is computed from the empirical distribution of the data. BI grows the working subset by adding other attributes into it one by one. At each step, it chooses the attribute that has the highest MI with the current subset. (The first intuition is used here.) The MI between a variable X and a set $\mathbf{S}$ is estimated as follows:

$$I(X;\mathbf{S}) = \max_{Z\in\mathbf{S}} I(X;Z) = \max_{Z\in\mathbf{S}} \sum_{X,Z} P(X,Z)\log\frac{P(X,Z)}{P(X)P(Z)} \tag{1}$$

BI determines when to stop expanding the working subset using the *unidimensionality test* or simply the *UD-test*. This is the most important idea of this paper. A subset of attributes is *unidimensional* if the optimal LTM (i.e., the one with the highest BIC score) for those attributes contains only one latent node. UD-test determines whether a subset of attributes is unidimensional by first projecting data onto those attributes and then running EAST on the projected data. If the resulting model contains only one latent variable, then UD-test concludes that the subset is unidimensional. Otherwise, it concludes the opposite. For computational efficiency, EAST is allowed to examine only models with 1 or 2 latent variables.

After each attribute is added to the working subset, BI runs the UD-test on the subset. If the test fails, the attributes in the subset cannot all be in one sibling cluster in the final model according to the second intuition. So, BI stops growing the subset and picks, from the local model learned during UD-test, the sibling cluster that contains (one of or both) the two initial attributes as the first sibling cluster for the whole algorithm. Attributes in the cluster are then removed from the data set and the process repeats to find other sibling clusters. Figure 2 illustrates the process. The working subset initially contains X_1 and X_2. Then X_3 is added. EAST is run to find an LTM for the expanded subset $\{X_1, X_2, X_3\}$. The resulting model, shown in (a), contains only one latent variable. So the triplet passes the UD-test. Then X_4 is added and the UD-test is again passed as shown in (b). After that X_5 is added. This time EAST yields a model, shown in (c), with more than one latent variable. So the UD-test fails and BI stops growing the subset. The sibling cluster $\{X_1, X_2, X_4\}$ of the local model is picked as the first sibling cluster for the whole algorithm.

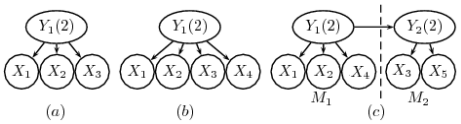


Fig. 2. An example of sibling cluster determination

Step 2: A *latent class model (LCM)* is an LTM with only one latent node. Figure 2 (a) and (b) show two examples. It is a commonly used finite mixture model

for discrete data. At Step 2, BI learns an LCM for each sibling cluster. There are two substasks: (1) to determine the cardinality of the latent variable and (2) to optimize the probability parameters. BI starts by setting the cardinality of the latent variable to 2 and optimizing the model parameters by running the EM algorithm [14]. Then it considers repeatedly increasing the cardinality. After each increase, model parameters are re-optimized. The process stops when the BIC score ceases to increase.

Step 3: After the first 2 steps, BI has obtained a collection of LCMs. We can visualize those LCMs as islands in an ocean. The next step is to link up the islands in a tree formation by adding edges between the latent variables.

Chow and Liu [15] give a well-known algorithm for learning tree-structured models among observed variables. It first estimates the MI between each pair of variables from data, then constructs a complete undirected graph with the MI values as edge weights, and finally finds the maximum spanning tree of the graph. The resulting tree model has the maximum likelihood among all tree models. Chow-Liu's algorithm can be adapted to link up the latent variables of the aforementioned LCMs. We only need to specify how the MI between two latent variables is to be estimated. Let m_1 and m_2 be two LCMs with latent variables Y_1 and Y_2. [3] Enumerate the data cases as $\mathbf{d}_1$, $\mathbf{d}_2$, ..., $\mathbf{d}_N$. Let $\mathbf{d}_{i1}$ and $\mathbf{d}_{i2}$ ($i \in \{1, 2, \ldots, N\}$) be respectively the projections of $\mathbf{d}_i$ onto the attributes of m_1 and m_2. Define the *data-conditional joint distribution* of Y_1 and Y_2 as follows:

$$P(Y_1, Y_2|\mathbf{D}, m_1, m_2) = C \sum_{i=1}^{N} P(Y_1|m_1, \mathbf{d}_{i1}) P(Y_2|m_2, \mathbf{d}_{i2}), \tag{2}$$

where C is the normalization constant. We estimate the MI between Y_1 and Y_2 from this joint distribution.

Step 4: In this step, BI optimizes the probability parameters of the LTM resulted from Step 3 using EM.

Step 5: The sibling clusters and the cardinalities of the latent variables were determined in Steps 1 and 2. Each of those decisions was made in the context of a small number of attributes. Now that all the variables are connected in a global model, it is time to re-examine the decisions and to see whether adjustments should be made.

In Step 5, BI checks each attribute to see whether it should be relocated and each latent variable to see if its cardinality should be changed. All the potential adjustments are evaluated with respect to the model, denoted by $\hat{m}$, resulted from the previous step. The beneficial adjustments are executed in one batch after all the evaluations. Adjustment evaluations and adjustment executions are not interleaved because that would require parameter optimization after each adjustment and hence be computationally expensive.

For each attribute X and each latent variable Y, BI computes their data-conditional MI $I(X, Y|\mathbf{D}, \hat{m})$ from the following distribution:

[3] Here m_1 and m_2 include model parameter values.

$$P(X, Y|\mathbf{D}, \hat{m}) = C \sum_{i=1}^{N} P(X, Y|\hat{m}, \mathbf{d}_i), \tag{3}$$

where C is the normalization constant. Let $\hat{Y}$ be the latent variable that has the highest data-conditional MI with X. If $\hat{Y}$ is not the current parent node of X in $\hat{m}$, then it is deemed beneficial to relocate X from its parent node to $\hat{Y}$.

To determine whether a change in the cardinality of a latent variable is beneficial, BI freezes all the parameters that are not affected by the change, runs EM locally to optimize the parameters affected by the change, and recalculates the BIC score. The change is deemed beneficial if the BIC is increased. BI starts from the current cardinality of each latent variable and considers increasing it by one. If it is beneficial to do so, further increases are considered.

After model refinement, EM is run on the global model one more time to optimize the parameters.

Computational Efficiency: BI is much faster than EAST. The reason is that most steps of BI involves only a small number of variables and the EM algorithm is run on the global model only twice. We tested BI and EAST on two data sets with 81/108 attributes and 3021/2763 samples respectively. EAST took 6/24 days, while BI took 35/69 minutes respectively. Detailed running time analysis will be given in a longer version of the paper.

4 Empirical Results with BI

We now present empirical results to demonstrate BI's capability in discovering rich and meaningful multiple clusterings. Comparisons with other methods will be made in the next section.

4.1 Clustering Synthetic Data

As a test of concept, we first tried BI on synthetic data. The data set was sampled from an LTM with structure as shown in Figure 3(left). There are 3 latent variables and 15 attributes, and all variables are binary. A total number of 1000 data cases were sampled. Each data case contains values for the attributes X_1-X_{15}, but not for latent variables Y_1-Y_3. The data set has 3 'true' partitions, each given by a latent variable. To provide some intuitions on what the partitions are about, Figure 3(right) depicts the normalized mutual information(NMI)[16] between each partition and each attribute. The x-axis represents the observed variables (i.e. X_1-X_{15}). The y-axis indicates the value of NMI between each latent variable(i.e. Y_1-Y_3) and each attribute. There are three curves, labeled as Y_1-Y_3. Those are called *feature curves* of the true partitions.

The LTM obtained by BI from the synthetic data has the same structure as the generative model. In particular, it has three latent variables. Each latent variable represents a soft partition of the data. We obtained a hard partition by assigning each data case to the state with the highest posterior probability. We call those partitions the *BI partitions*. The curves labeled B_1-B_3 in Figure 3 are

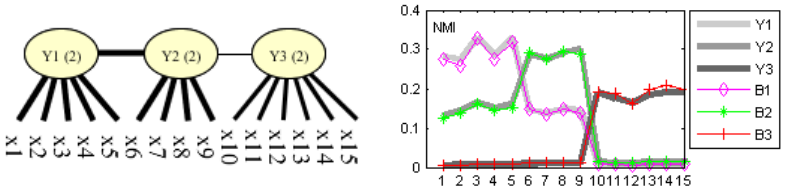


Fig. 3. Generative model and feature curves of partitions.

the feature curves of the BI partitions. They match those of the true partitions almost perfectly. Based on the feature curves, we matched up the BI partitions with the true partitions, and computed the NMI for each pair. The NMI values are 0.91(B1,Y1), 0.86(B2,Y2) and 0.98(B3,Y3) respectively. Those indicate that BI has recovered the true partitions well.

4.2 Clustering Real-World Data

The real-world data set is known as WebKB data. It consists of web pages collected in 1997 from the computer science departments of 4 universities: Cornell, Texas, Washington and Wisconsin. The web pages were originally divided into 7 classes. Only 4 classes remain after preprocessing, namely student, faculty, project and course. There are 1041 pages in total and the number of words was reduced to 336. The words are used as attributes in our analysis. The attributes are binary and indicate the presence or absence of the words. Both the university labels and class labels were removed before the analysis.

On the WebKB data set, BI produced an LTM, henceforth called *WebKB-LTM*, with 98 latent variables. This means that BI has partitioned the data in 98 different ways. A big question is: Are those partitions meaningful or do they appear to be arbitrary? It turns out that many of the partitions are meaningful. We present some of them in the following.

To start with, we give an overview of the structure of WebKB-LTM. It divides itself into three parts. The model of the first part is shown in Figure 1. It involves words that usually appear in general descriptions of computer science departments. So we call it the *department subnet*. The second part contain words such as research, interest, papers, ieee, etc. We call it the *research subnet*. The third part involves words related to course teaching. So we call it the *teach subnet*.

The Department Subnet: We first examine two latent variables Y_{51} and Y_{54} in Figure 1. To inspect the meanings of partitions, we can draw the information curves as shown in Figure 4. Take information curves of Y_{51} as example, there are two curves in the figure. The lower one shows the mutual information(MI) between Y_{51} and each attribute. The attributes are sorted according to MI and only the top 5 attributes are shown here. The upper curve shows the cumulative mutual information(CMI) between Y_{51} and each attribute plus all the attributes before it. The numbers on the left vertical axis are MI values. The numbers on the right axis are the ratios of the MI values over the MI between the partition and all the attributes. The ratio is called *information coverage (IC)*. The IC of

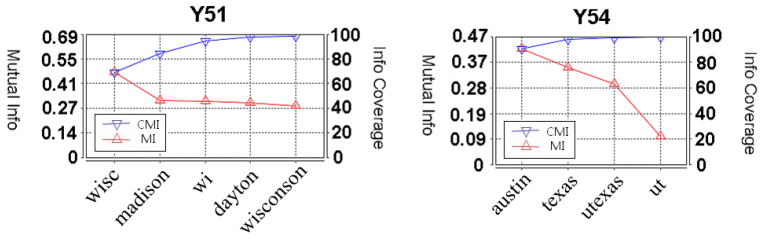


Fig. 4. Information curves of latent variables Y_{51} and Y_{54}.

the first 5 attributes is around 98%. Intuitively, this means that the differences among the different clusters on the first 5 attributes account for 98% of the total differences. So we can say that the partition is primarily based on these attributes.

The table below shows the occurrence frequencies of the words in the clusters. We see that Y_{51} represents a partition based on wisc, madison, wi, dayton and wisconsin. The clusters Y_{51}=2 and Y_{51}=3 together correspond to U Wisconsin-Madison. Y_{54} represents another partition based on austin, texas, utexas and ut. The cluster Y_{54}=2 corresponds to U Texas-Austin.

Y_{47}: IC=94%

cluster	1	2
cornell	.04	1
ithaca	0	.47
ny	.01	.39
Size	.84	.16

Y_{48}: IC=94%

cluster	1	2	3
washington	.03	1	.98
seattle	.01	.13	1
wa	0	0	.92
Size	.8	.1	.1

Y_{51}: IC=98%

cluster	1	2	3
wisc	0	.86	.96
madison	.01	.33	.98
wi	0	.03	.92
dayton	0	.04	.92
wisconsin	0	.31	.94
Size	.74	.14	.12

Y_{54}: IC=99%

cluster	1	2
austin	0	.86
texas	.01	.33
utexas	0	.03
ut	0	.04
Size	.82	.18

We can do similar analysis to Y_{47} and Y_{48}. As shown in above tables, the most informative attributes are selected based on information curves, which are omitted to save space. Information coverages (IC) are given to show how well the attributes collectively convey the meanings of the latent variables. It is clear that Y_{47} and Y_{48} are also meaningful. Y_{47}=2 seems to identify web pages from Cornell, and Y_{48}=2 and Y_{48}=3 together seem to identify web pages from U Washington-Seattle.

The Research Subnet: The following tables contain information about three latent variables from the faculty subnet.

Y_{10}: IC=92%

cluster	1	2
image	.02	.5
images	.02	.38
visual	.01	.31
pattern	0	.19
vision	.02	.3
Size	.91	.09

Y_{11}: IC=90%

cluster	1	2
management	.02	.57
database	.04	.58
storage	0	.29
databases	.01	.33
query	0	.3
Size	.88	.12

Y_{14}: IC=95%

cluster	1	2
april	.02	.42
march	.01	.35
february	0	.33
conference	.06	.44
symposium	.03	.33
proceedings	.04	.38
january	.02	.31
international	.05	.32
Size	.86	.14

Latent variable Y_{10} partitions the web pages based on words such as image, pattern and vision. Y_{10}=2 seems to identify web pages belonging to faculty members who work in the vision/image analysis area. Y_{11} partitions the web pages based on words such as database, storage and query. Y_{11}=2 seems to identify web pages belonging to faculty members who work in the database area. Other

latent variables in the faculty subset give us clusters of web pages for other research areas such as artificial intelligence, networking, etc. Besides research area-related partitions, the faculty subnet also gives us some other interesting clusterings. One example is Y_{14}. Y_{14}=2 seems to identify web pages containing papers published at conferences held in January-April.

The Teach Subnet: The teach subnet gives us partitions based on various aspects of course information. One example is Y_{66}. It is clear from the information given below, Y_{66}=4 identifies web pages on objected-oriented programming (OOP), while Y_{66}=2 identifies web pages on programming but not mentioning OOP. Those might be web pages of OOP courses and of introductory programming courses respectively. Y_{66}=3 seems to correspond to web pages of other courses that involve programming, while Y_{66}=1 seems to mean web pages not on programming.

Y_{66}: IC=94%

cluster	1	2	3	4
programming	.05	.77	1	.71
oriented	0	0	.21	1
object	.02	.06	.46	.91
language	.05	.32	.86	.59
languages	.01	.39	.22	.48
program	.09	.26	.96	.27
programs	.04	.2	.6	.16
Size	.69	.21	.05	.05

In summary, the BI algorithm has identified a variety of interesting clusters in the WebKB data. The situation is similar on several other data sets that we tried. It should be noted that not all the results obtained by BI are meaningful and interesting. After all, it is an explorative data analysis method.

5 Comparisons with Alternative Methods

In this section we compare BI with four other MPC algorithms: orthogonal projection (OP) [1], singular alternative clustering (SAC) [2], DK [4] and EAST. Those are the algorithms that we were able to obtain implementations for or implement ourselves.

The motivation for MPC is the observation that complex data can be meaningfully clustered in multiple ways. As such, we need to consider two questions when comparing different MPC methods: (1) Do they find meaningful clusterings? (2) How many meaningful clusterings do they find?

5.1 Meaningful Clustering

A common way to evaluate a single-partition clustering algorithm is to start with labeled data, remove the class labels, perform cluster analysis, and compare the partition obtained with the partition induced by the class labels. We refer to those two partitions as the *cluster partition* and the *class partition* respectively. The quality of the cluster partition is often measured using its NMI with the class partition.

In the context of MPC, we have multiple class partitions and multiple cluster partitions. How should the evaluation be carried out? Following the literature,

we match each class partition up with the cluster partition with which it has the highest NMI and report the NMI values of the matched pairs. This means that if one of the cluster partitions closely resemble a class partition, then we claim that the class partition is recovered from data. Similar partitions have similar feature curves. So, we sometimes can do the match up based on feature curves, as was done in Section 4.1. A nice thing with the use of feature curves is that they tell us what the partitions are about.

Results on Synthetic Data: We tested OP, SAC, DK and EAST on the same synthetic data that was used to test BI in Section 4.1. The published version of DK can only produce two partitions. We ran DK on the synthetic data to obtain two, instead of three, binary partitions. OP and SAC are sequential MPC methods. Following the authors, we first ran K-means to find a partition of two clusters, and then continue to run OP and SAC to find two more binary partitions.

We have run the experiments 10 times for each algorithm. Here are the NMI values between each true class partition and the matched cluster partition obtained by the algorithms:

	DK	SAC	OP	EAST	BI
Y1	.78±.00	.73±.04	.73±.02	**.91±.00**	**.91±.00**
Y2	.48±.00	.57±.04	.55±.04	**.86±.00**	**.86±.00**
Y3	.97±.00	.59±.49	.91±.20	**.98±.00**	**.98±.00**

The NMI values are high for EAST and BI, indicating that those two algorithms have recovered the three true class partitions well. On the other hand, the NMI values for the other algorithms are relatively lower, indicating that they have not been able to recover the true class partitions as well as EAST and BI.

The results by EAST and BI are identical in terms of NMI values. However, BI took much less time than EAST. The running time of BI was 22±3 seconds, while that of EAST was 485±34 seconds.

Results on Real-World Data: DK, OP and SAC were also tested on the WebKB data. They were told to find two partitions each with four clusters, because it is known that there are two true class partitions each with four classes.

One of class partition divides the web pages into four classes according to the four universities. It is clear from Section 4.2 that BI has recovered the four classes. However, they were given in the form of four partitions (Y_{47}, Y_{48}, Y_{51} and Y_{54}), instead of one. For comparability, we transformed the 4-class class partition into four logically equivalent binary class partitions. Each binary class partition divides the web pages according to whether they are from a particular university. The same transformation was applied to the other class partition and the cluster partitions obtained by the alternative algorithms. After the transformations, we matched up the binary class partitions with the cluster partitions and computed the NMI of each matched pair. The results are shown in following tables. The average was taken over 10 runs of the algorithms.

We see that the NMI is the highest for BI in almost all cases. This means that BI has recovered most of the 8 classes better than the alternative algorithms.[4]

[4] As seen in Section 4.2, some of the partitions obtained by BI contain multiple clusters that correspond to a true class. For example, both $Y_{48} = 2$ and $Y_{48} = 3$ correspond

	DK	SAC	OP	BI
course	.43±.01	.47±.01	.47±.02	**.59±.01**
faculty	.18±.04	.17±.07	.18±.01	**.31±.01**
project	.04±.00	.04±.00	.05±.04	**.07±.00**
student	.18±.00	**.20±.00**	**.20±.01**	**.20±.02**
cornell	.22±.15	.09±.02	.36±.24	**.48±.11**
texas	.31±.18	.20±.20	.45±.23	**.60±.02**
washington	.22±.13	.41±.23	**.56±.25**	.52±.12
wisconsin	.38±.12	.16±.12	.45±.13	**.48±.10**

We also tested EAST on WebKb data. However, it did not finish in two weeks. In contrast, BI took only 90 minutes.

5.2 Richness of Clusterings

The WebKB data was analyzed by a number of authors using different methods [1, 5, 7]. In all cases, only two partitions and a few clusters were obtained. In contrast, BI has discovered , as shown in Section 4.2, a rich collection of meaningful clusters. This is what sets BI far apart from other methods.

Why is it difficult for other methods to produce rich clustering results on complex data sets? Well, the sequential MPC methods and the distance-based simultaneous MPC methods cannot determine the numbers of partitions and clusters. The user has to provide such information as input. In single partitioning, not being able to determine the number of clusters is not a big problem. One can manually try a few possibilities and pick the one that yields the best results. In the case of MPC, however, not being able to determine the numbers of partitions and clusters is a severe drawback. There are too many possibilities to consider. As a simplification of the problem, suppose it is known that there are 10 partitions. Determining the number of clusters for the 10 partitions manually would be very difficult as there are too many possible combinations to try.

Other model-based simultaneous MPC methods can determine the numbers of partitions and clusters. However, they are still unable to produce rich clustering results on complex data sets. The algorithm by Galimberti and Soffiritti [6] is inefficient and has so far been tested only on data sets with fewer than 10 attributes. The EAST algorithm is not efficient enough to handle data sets with 100 or more attributes. The method by Guan et al. [7] is efficient enough to deal with the WebKb data, but it produces only two partitions.

5.3 A Remark

The evaluation method used in Section 5.1 has one drawback. A naïve method that generates a huge number of random partitions might get good evaluation. So, it is important to test MPC methods on real-world data set and see whether they can help find meaningful clusters. On the WebKB data, BI found 98 partitions. By inspecting their information curves, we were able to quickly identify dozens of meaningful partitions. With the random method, however, one would have to examine a huge number of partitions before finding a meaningful one. Hence it is not useful.

to U Washington-Seattle. If such clusters are manually aggregated, the NMI values for BI would look better.

6 Conclusions

This paper is concerned with the use of latent tree models (LTMs) for multiple partition clustering. We propose a new algorithm, called BI, for learning LTM that is much faster than the best previous algorithm and can handle data with hundreds of attributes. The key idea behind the algorithm is UD-test, which determines whether the interactions among a set of observed variables can be modeled using one latent variable. Empirical results are presented to show that BI is able to produce much richer clustering results than alternative methods.

7 Acknowledgements

Research on this work was supported by Hong Kong Research Grants Council GRF Grant #622408, The National Basic Research Program of China (aka the 973 Program) under project No. 2011CB505101, and HKUST Fok Ying Tung Graduate School.

References

1. Cui, Y., Fern, X.Z., Dy, J.G.: Non-reduntant multi-view clustering via orthogonalization. In: ICDM-07. (2007)
2. Qi, Z., Davidson, I.: A principled and flexible framework for finding alternative clusterings. In: KDD-09. (2009)
3. Gondek, D., Hofmann, T.: Non-redundant data clustering. KAIS-07 (1) (2007)
4. Jain, P., Meka, R., Dhillon, I.S.: Simultaneous unsupervised learning of disparate clusterings. In: SDM-08. (2008) 858–869
5. Niu, D., Dy, J.G., Jordan, M.I.: Multiple non-redundant spectral clustering views. In: ICML-10. (2010)
6. Galimberti, G., Soffritti, G.: Model-based methods to identify multiple cluster structures in a data set. CSDA-07 **52** (2007) 520–536
7. Guan, Y., Dy, J.G., Niu, D., Ghahramani, Z.: Variational inference for nonparametric multiple clustering. In: MultiClust Workshop, KDD-2010. (2010)
8. Zhang, N.L.: Hierarchical latent class models for cluster analysis. JMLR-04 **5** (2004) 697–723
9. Poon, L.K.M., Zhang, N.L., Chen, T., Yi, W.: Variable selection in model-based clustering: To do or to facilitate. In: ICML-10. (2010)
10. Schwarz, G.: Estimating the dimension of a model. The Annals of Statistics **6**(2) (1978) 461–464
11. Chen, T., Zhang, N.L., Wang, Y.: Efficient model evaluation in the search-based approach to latent structure discovery. In: PGM-08, 57-64. (2008)
12. Harmeling, S., Williams, C.K.I.: Greedy learning of binary latent trees. TPAMI-10 (2010)
13. Choi, M.J., Tan, V.Y.F., Anandkumar, A., Willsky, A.S.: Learning latent tree graphical models. Computing Research Repository (2010)
14. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
15. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory (1968)
16. Strehl, A., Ghosh, J.: Cluster ensembles — a knowledge reuse framework for combining multiple partitions. JMLR **3** (2002) 583–617

Factorial Clustering with an Application to Plant Distribution Data

Manfred Jaeger[1], Simon P. Lyager[1], Michael W. Vandborg[1], and Thomas Wohlgemuth[2]

[1] Dept. of Computer Science, Aalborg University, Denmark
[2] Swiss Federal Research Institute WSL

Abstract. We propose a latent variable approach for multiple clustering of categorical data. We use logistic regression models for the conditional distribution of observable features given the latent cluster variables. This model supports an interpretation of the different clusterings as representing distinct, independent factors that determine the distribution of the observed features. We apply the model for the analysis of plant distribution data, where multiple clusterings are of interest to determine the major underlying factors that determine the vegetation in a geographical region.

1 Introduction

There exist a variety of different approaches to learning multiple clusterings. They can differ not only with regard to their mathematical models and algorithmic methods, but there can also be widely different intuitions and objectives with regard to the interpretation of the multiple clusterings. On the one hand, in ensemble clustering, the individual clusterings are essentially regarded as different, imperfect versions of a single underlying true clustering (e.g. [10]). In many multiple clustering methods, on the other hand, the different clusterings are intended to represent different views of the data, each providing a different insight into the structure of the data. One objective for clustering methods then is to ensure that different clusterings are in some sense independent, disparate [6], or non-redundant [8].

Probabilistic latent variable models are used for a variety of data analysis tasks (in the case of discrete data jointly referred to as *latent class analysis*), including clustering. Several authors have investigated probabilistic latent variable models for multiple clusterings [13, 4, 2]. For the case of discrete observable features, no special assumptions on the distributional form of the features given the latent variables are made in these approaches, i.e. the conditional distribution of the features follows an unconstrained multinomial distribution. Latent variable models are also commonly used for dimensionality reduction of high-dimensional numeric data. An important example is the *factor analysis* model, in which the observed data is interpreted as a noisy linear transformation of a small number of latent dimensions. While it is quite common to refer to latent class analysis

as a "categorical data analogue to factor analysis" [5][1, Chapter 13], it seems that this correspondence has not been fully exploited for clustering applications, or put into the context of multi-clustering, via the combined use of multiple latent variables, and special assumptions on the conditional distribution of the observed features.

In this paper we propose a probabilistic latent variable model for multiple clusterings. As in factor analysis, we interpret the observed (discrete) data as a noisy transformation of underlying, discrete latent dimensions. The linear mapping of factor analysis is replaced by a log-linear logistic regression model. The latent dimensions then define clusterings that can be seen as independent factors that determine the distribution of the observed features.

In contrast with several other multiple clustering methods (e.g., [4, 8]) our method is not based on an association of different clusterings with different feature subsets, even though such associations can emerge.

Our approach is partly motivated by applications to biogeographical data. Specifically, we are investigating plant distribution data. Segmentations of geographic units into floristic regions based on similarity of plant species composition were already undertaken in the 19th century. An early application of formal methods of clustering in this context is [9]. We apply our method to distribution data for 2398 plant species in Switzerland. The goal of factorial clustering for this type of data will be to obtain multiple clusterings, each of which could correspond to one of several underlying environmental, geographical, or historical factors, which jointly influence the vegetation.

2 Latent Variable Models for Clustering

Latent variable models are routinely used for clustering, both for single and multiple clustering. However, they can be used in several, slightly different ways. In order to more clearly explain our approach, we briefly review in this section possible approaches to using latent variable models for clusterings.

Throughout, we assume that the observable data $\boldsymbol{X}$ consists of n observations of k attributes, i.e. $\boldsymbol{X}$ is an $n \times k$ matrix. A latent variable model contains m additional unobserved variables, and we denote with $\boldsymbol{L}$ the $n \times m$ matrix of the latent variables in the n observations. We note that when we assume that in the n observations both the observable and latent variables are identically and independently sampled, it will be simpler and more natural to describe the model in terms of vectors $\boldsymbol{X}$, $\boldsymbol{L}$ of length n and m, respectively. However, in some applications, especially segmentation of time sequences or images, the latent variables are not independent at different data points.

A latent variable model, then, consists of a joint distribution for $\boldsymbol{X}$ and $\boldsymbol{L}$, which can be written as

$$P(\boldsymbol{X} \mid \boldsymbol{L}, \theta_{\boldsymbol{X}|\boldsymbol{L}}) P(\boldsymbol{L} \mid \theta_{\boldsymbol{L}}). \tag{1}$$

In hierarchical models, this might be extended by a distribution over $\theta_{\boldsymbol{X}|\boldsymbol{L}}, \theta_{\boldsymbol{L}}$ parametrized by hyperparameters $\boldsymbol{\lambda}$.

The perhaps most common use of model (1) for clustering is to perform two steps [13]: first, fit the parameters $\theta_{\boldsymbol{X}|\boldsymbol{L}}, \theta_{\boldsymbol{L}}$ by maximizing the marginal likelihood of the observed data $\boldsymbol{X} = \boldsymbol{x}$:

$$(\theta^*_{\boldsymbol{X}|\boldsymbol{L}}, \theta^*_{\boldsymbol{L}}) := \underset{\theta_{\boldsymbol{X}|\boldsymbol{L}}, \theta_{\boldsymbol{L}}}{arg\,max} \sum_{\boldsymbol{l}} P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{L} = \boldsymbol{l}, \theta_{\boldsymbol{X}|\boldsymbol{L}}) P(\boldsymbol{L} = \boldsymbol{l} \mid \theta_{\boldsymbol{L}}). \tag{2}$$

This step is usually performed using the EM algorithm. Then, compute the most probable values of $\boldsymbol{L}$ given $\boldsymbol{X} = \boldsymbol{x}$:

$$\boldsymbol{l}^* = \underset{\boldsymbol{l}}{arg\,max}\, P(\boldsymbol{L} = \boldsymbol{l} \mid \boldsymbol{X} = \boldsymbol{x}, \theta^*_{\boldsymbol{X}|\boldsymbol{L}}, \theta^*_{\boldsymbol{L}}) \tag{3}$$

In multiple clustering, a joint configuration of the latent variables defines multiple cluster indices. For simplicity we may assume for now that each latent variable defines its own clustering, and that therefore the membership of the ith data item in the jth clustering is given by $\boldsymbol{l}^*_{i,j}$. However, in the multi-cluster case, the second step can also take a slightly different form, and the most probable latent variable values be computed component-wise. Denoting by $\boldsymbol{l}_j$ the jth column of $\boldsymbol{l}$ (i.e., $\boldsymbol{l} = (\boldsymbol{l}_1, \ldots, \boldsymbol{l}_m)$), this can be written as

$$\boldsymbol{l}^*_j = \underset{\boldsymbol{l}_j}{arg\,max} \sum_{\boldsymbol{l}_1, \ldots, \boldsymbol{l}_{j-1}, \boldsymbol{l}_{j+1}, \boldsymbol{l}_m} P(\boldsymbol{L} = \boldsymbol{l} \mid \boldsymbol{X} = \boldsymbol{x}, \theta^*_{\boldsymbol{X}|\boldsymbol{L}}, \theta^*_{\boldsymbol{L}}). \tag{4}$$

This is the (hard) clustering rule used, e.g., in [13, 14]. The clusterings obtained from (3) and (4) can differ.

If the ultimate goal is only to compute a most probable configuration of $\boldsymbol{L}$, then one may also try to simplify the combination of (2) and (3) into a single optimization:

$$\boldsymbol{l}^* := \underset{\boldsymbol{l}}{arg\,max} \underset{\theta_{\boldsymbol{X}|\boldsymbol{L}}, \theta_{\boldsymbol{L}}}{max} P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{L} = \boldsymbol{l}, \theta_{\boldsymbol{X}|\boldsymbol{L}}) P(\boldsymbol{L} = \boldsymbol{l} \mid \theta_{\boldsymbol{L}}). \tag{5}$$

This rule can be justified by a Bayesian interpretation, for example: it amounts to finding the jointly most probably values of $\boldsymbol{l}, \theta_{\boldsymbol{X}|\boldsymbol{L}}, \theta_{\boldsymbol{L}}$, given the data $\boldsymbol{X} = \boldsymbol{x}$, and assuming a uniform prior for $\theta_{\boldsymbol{X}|\boldsymbol{L}}, \theta_{\boldsymbol{L}}$. Rule (5) may be still further simplified, if one assumes the model for the latent variables to be fixed, and not subject to optimization, i.e., $P(\boldsymbol{L} = \boldsymbol{l} \mid \theta_{\boldsymbol{L}}) = P(\boldsymbol{L} = \boldsymbol{l} \mid \theta^*_{\boldsymbol{L}})$ for fixed parameters $\theta^*_{\boldsymbol{L}}$, and the parameter optimization is only for $\theta_{\boldsymbol{X}|\boldsymbol{L}}$. If, furthermore, $P(\boldsymbol{L} = \boldsymbol{l} \mid \theta^*_{\boldsymbol{L}})$ is assumed uniform, then (5) reduces to

$$\boldsymbol{l}^* := \underset{\boldsymbol{l}}{arg\,max} \underset{\theta_{\boldsymbol{X}|\boldsymbol{L}}}{max} P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{L} = \boldsymbol{l}, \theta_{\boldsymbol{X}|\boldsymbol{L}}). \tag{6}$$

Whether it is justified to assume a fixed distribution $P(\boldsymbol{L} \mid \theta^*_{\boldsymbol{L}})$ can depend on two considerations: first, assuming that (1) actually represents the generative process for the data, one might have sufficient background knowledge to identify the distribution of $\boldsymbol{L}$ a-priori. $\boldsymbol{L}$ being an unobserved variable, whose existence is essentially hypothesized, and for which it is typically even unclear how many

states it has, this is a rather unlikely case in practice, however. Second, clustering being an exploratory data-analysis tool, one may also consider what settings of $P(\boldsymbol{L} \mid \theta^*_{\boldsymbol{L}})$ may lead via (5) to interesting insights into the data, regardless of whether the underlying probabilistic model is accurate as a generative model.

For example, in the single clustering case, when the data is generated by a mixture model where one mixture component has a much higher prior probability than the others, then clustering via (3) can easily lead to only obtaining a single cluster. If, on the other hand, one eliminates the influence of the prior distribution by assuming (incorrectly) a uniform distribution over the mixture components, then clustering via (6) can reveal the mixture structure of the data.

3 The Factorial Logistic Model

In the factor analysis model, both $\boldsymbol{X}$ and $\boldsymbol{L}$ are numerical, the rows in $\boldsymbol{X}$ and $\boldsymbol{L}$ are iid, and the model (1) is given by distribution

$$P(\boldsymbol{L}_i) \sim N(\boldsymbol{0}, \boldsymbol{\Sigma_L})$$
$$P(\boldsymbol{X}_i \mid \boldsymbol{L}_i) \sim N(\boldsymbol{W}\boldsymbol{L}_i + \boldsymbol{\mu}, \boldsymbol{\Sigma_X})$$

where $\boldsymbol{\Sigma_L}$ is an arbitrary covariance matrix, $\boldsymbol{W}$ is a $k \times m$ matrix, $\boldsymbol{\mu}$ a m-dimensional mean vector, and $\boldsymbol{\Sigma_X}$ a diagonal covariance matrix. Thus, data is assumed to be generated by sampling from a lower (k) dimensional Gaussian distribution, linearly mapped into the higher (m) dimensional space, and independent Gaussian noise added to each coordinate.

The logistic regression model for the distribution of a binary variable X conditional on numeric latent variables $\boldsymbol{L}$ is given by

$$\log P(X = 1 \mid \boldsymbol{L})/P(X = 0 \mid \boldsymbol{L}) = w_0 + \boldsymbol{w}\boldsymbol{L}, \tag{7}$$

where $\boldsymbol{w} = (w_1, \ldots, w_k)$ is a k-vector of real weights. We write $X \sim LR(w_0, \boldsymbol{w}\boldsymbol{L})$ if X follows (7). This model also applies when the latent variables $\boldsymbol{L}$ are *ordinal*, i.e. each L_j codes by an integer $\{0, \ldots, r_j - 1\}$ one of r_j different, ordered categories. To accommodate *nominal* predictor variables (i.e., unordered categorical variables) in the logistic regression model, one encodes a nominal variable L_j with r states by binary indicator variables $L_{j,1}, \ldots, L_{j,r}$, i.e. $L_{j,h} = 1$, $L_{j,h'} = 0$ ($h' \neq h$) means that L_j is in its hth state.

We will consider both ordinal and nominal latent variables for clustering. An ordinal latent variable defines an ordered clustering, i.e. the cluster indices define an ordering of the clusters. Whether such an ordering is meaningful and interpretable is application dependent. For biogeographical data ordinal latent variables and ordered clusterings are often natural, since data patterns are often determined by underlying continuous variables. We will, thus, assume that $\boldsymbol{L}$ is a vector of m latent variables that define c different clusterings. Furthermore, we assume that one of the following two cases applies: (1) all L_j in $\boldsymbol{L}$ are ordinal; in this case $c = m$, and the jth clustering consists of r_j distinct cluster. (2) $\boldsymbol{L}$ is an encoding by binary indicator variables of c distinct nominal variables

with $r_1, \ldots, r_c$ distinct states, respectively. In this case $m = \sum_{i=1}^{c} r_i$. We refer to model (1) as the $(r_1 o, \ldots, r_k o)$ model, and (2) as the $(r_1 n, \ldots, r_c n)$ model. One could also consider models combining ordinal and nominal latent variables, but we will here focus on "pure" models.

As in the factor analysis model, we assume that $P(\boldsymbol{X} \mid \boldsymbol{L}) \sim \prod_{i=1}^{n} P(\boldsymbol{X}_i \mid \boldsymbol{L}_i) \sim \prod_{i=1}^{n} \prod_{j=1}^{k} P(\boldsymbol{X}_{i,j} \mid \boldsymbol{L}_i)$. Assuming that each $X_{i,j}$ follows a logistic regression model (7) with parameters $w_{j,0}, \boldsymbol{w}_j$, one obtains the model for the ith data item:

$$P(\boldsymbol{X}_i \mid \boldsymbol{L}_i) \sim \prod_{j=1}^{k} LR(w_{j,0}, \boldsymbol{w}_j \boldsymbol{L}_i). \tag{8}$$

This conditional model for $\boldsymbol{X}$ may be combined with various models for $P(\boldsymbol{L})$, with or without an iid assumption for the rows of $\boldsymbol{L}$. We refer to multiple clustering based on (8) as *factorial logistic (FL)* clustering.

4 Learning

We apply the simple learning rule (6) for clustering with the logistic regression model. Thus, we assume that $\boldsymbol{L}$ is uniformly distributed, which implies, in particular, independence over rows: $P(\boldsymbol{L}) \sim \prod_i P(\boldsymbol{L}_i)$. In case of $\boldsymbol{L}$ encoding nominal variables, the uniform distribution, of course, is conditional on "legal" states of $\boldsymbol{L}$, i.e. at most one indicator variable for any particular nominal variable being equal to 1.

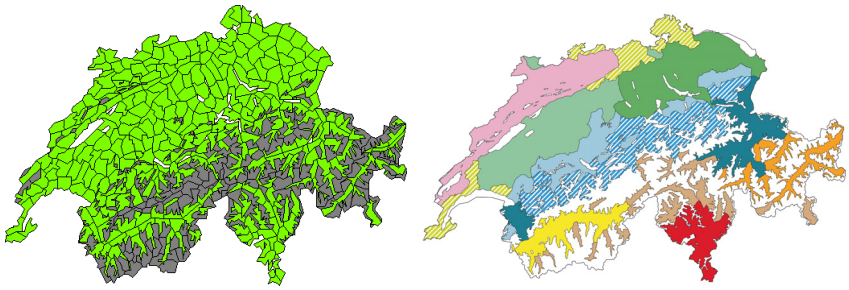
For the optimization of (6) we then use the obvious iterative procedure, where after a random initialization $\boldsymbol{L} := \boldsymbol{l}_0$ two steps are alternated:

i $\theta_{\boldsymbol{X}|\boldsymbol{L}_t} := arg\,max_{\theta_{\boldsymbol{X}|\boldsymbol{L}}} P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{L} = \boldsymbol{l}_t, \theta_{\boldsymbol{X}|\boldsymbol{L}})$
ii $\boldsymbol{l}_{t+1} := arg\,max_{\boldsymbol{l}} P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{L} = \boldsymbol{l}, \theta_{\boldsymbol{X}|\boldsymbol{L}_t})$

Step **i** is performed in our implementation using the SPSS method of fitting logistic regression models, which supports both ordinal and nominal predictor variables. Due to the factorization (8), the optimization reduces to k independent optimizations for the parameters $(w_{j,0}, \boldsymbol{w}_j)$ $(j = 1, \ldots, k)$. It is thus linear in k. It also is linear in n, since the likelihood only depends on the counts $|\{i \mid \boldsymbol{X}_{i,j} = 1, \boldsymbol{L}_i = \hat{\boldsymbol{l}}\}|$ for fixed configurations $\hat{\boldsymbol{l}}$ of the latent variables.

For step **ii** we have $P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{L} = \boldsymbol{l}, \theta_{\boldsymbol{X}|\boldsymbol{L}_t}) = \prod_i P(\boldsymbol{X}_i = \boldsymbol{x}_i \mid \boldsymbol{L}_i = \boldsymbol{l}_i, \theta_{\boldsymbol{X}|\boldsymbol{L}_t})$, so that the problem decomposes into n distinct optimizations for the $\boldsymbol{l}_i$. It can be naively performed by computing $P(\boldsymbol{X}_i = \boldsymbol{x}_i \mid \boldsymbol{L}_i = \boldsymbol{l}_i, \theta_{\boldsymbol{X}|\boldsymbol{L}_t})$ for each candidate $\boldsymbol{l}_i$, which gives a procedure that is still linear in n and k, but exponential in c.

Overall, we obtain a learning method that is linear in the number of data items and the observable attributes, and exponential in the number of clusterings.



(a) (b)

Fig. 1. Mapping areas with mountain - valley division (a), and previous segmentation of valley areas into floristic regions [12] (b)

5 Experiments

We apply FL-clustering to geobotanical data. In our experiments we use the source data for the "Swiss Web Flora" [3] [11]. The dataset contains information on the distribution of 2697 plant species in Switzerland, which has been divided into 565 mapping areas. We reduced slightly more detailed species abundance information in the original data to simple binary presence/absence data. We also in this process deleted plants with a very sparse and uncertain distribution. This left us with 2398 species in our data.[4] We view each plant species as an observable attribute, and the mapping areas as independent observations. Thus, $n = 565$ and $k = 2398$ in the notation of Section 2. Figure 1 shows the division of Switzerland into the mapping areas. Apart from the species occurrence data, only a single additional variable is recorded for each area: a binary variable that indicates whether the area is a mountain area (above timberline), or a valley area (below timberline). The value of this variable is shown if Figure 1 by a green color for valley, and grey color for mountain areas.

Conventional (single-) clusterings of the data lead to a segmentation of Switzerland into *floristic regions*. Figure 1 (b) shows a result obtained by agglomerative hierarchical clustering of the valley areas only [12] (thus, the white part of the figure does not correspond to a computed cluster; it comprises areas not included in the clustering).

5.1 Synthetic Data

In order to obtain an initial evaluation of the feasibility of our approach, we first conduct an experiment with synthetic data. For this we constructed two artificial segmentations of Switzerland based on the same mapping regions as in the real

[3] `www.wsl.ch/land/products/webflora/welcome-en.ehtml`

[4] The data is available at http://www.wsl.ch/info/mitarbeitende/wohlgemu/lehre_EN/

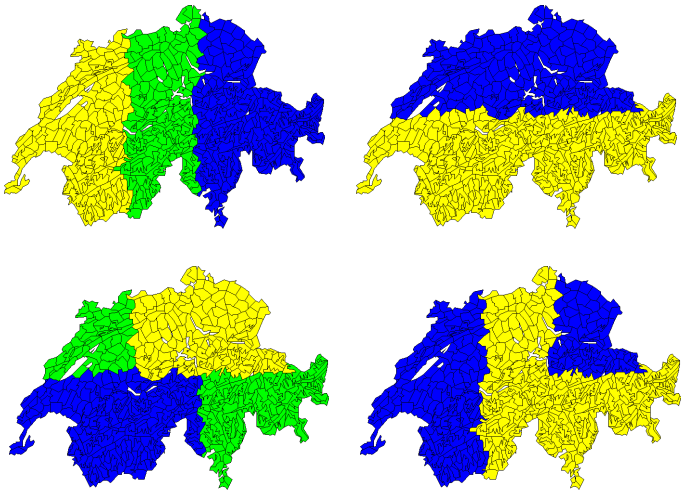


Fig. 2. Artificial segmentations (top); "Wrong" clusterings (bottom)

data. These segmentations are shown in Figure 2 (top), and henceforth referred to as "vertical" and "horizontal" segmentation, respectively. For each combination of a vertical and a horizontal segment, we defined a species distribution type by a nominal logistic regression model that expresses a preference of the species for the selected vertical and horizontal segment. The logistic regression weights were adjusted so as to obtain conditional probability distributions for the presence of a species of the following form (here showing the case of preference for the first segment in both segmentations):

	Vertical		
Horizontal	yellow	green	blue
blue	0.98	0.5	0.5
yellow	0.5	0.02	0.02

(9)

According to each distribution type we created 15 synthetic plant species, and randomly sampled an occurrence variable for the species at each of the mapping areas.

We then performed FL clustering based on the 90 synthetic species using the (3o,2o) model (it is not our ambition at this point to detect the "right" number of segmentations and segments per segmentation). In approximately 1 out of 3 random restarts the algorithm terminated with the correct segmentations of Figure 2, or solutions that differed from the correct one in cluster assignments for 2-3 regions. In the remaining restarts the algorithm terminated at local optima, a representative example of which is shown in Figure 2 (bottom). However, the (almost) correct solutions were identified by higher log-likelihood scores (between -19912 and -19783) than that of the wrong solutions (between -23474 and -21677).

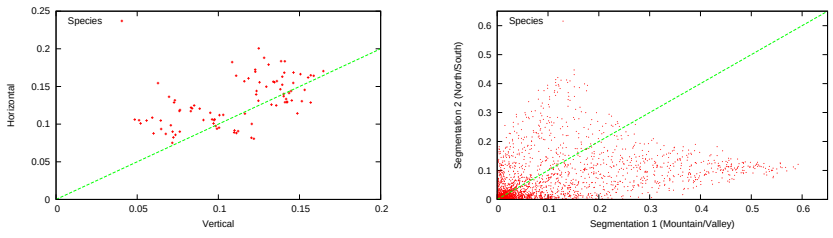
For comparison, we also performed an experiment where the logistic regression model for $P(\boldsymbol{X} \mid \boldsymbol{L})$ was replaced by a full multinomial model, i.e. for each species we fit a conditional probability table of the form (9) with 6 independent parameters. In this case, almost all restarts terminated with wrong solutions as in Figure 2, and, more importantly, the correct solutions could not be distinguished by a higher likelihood score: in the multinomial model, any pair of segmentations whose combination identifies the 6 different combinations of vertical and horizontal segments achieves the same, optimal, likelihood score.

We also use this synthetic data experiment to demonstrate that in FL-clustering there is not necessarily a correlation between clusterings and feature-subsets. Figure 3 (a) shows for each of the 90 synthetic species the mutual information between the species occurrence feature and the two clusterings of Figure 2 (top). The plot shows that there is no strong association of individual species features with one or the other of the two segmentations.

5.2 Real Data

We now perform experiments with the real data consisting of the actual 2398 species. Again, we do not try at this point to automatically detect an appropriate number of segmentations, or segments per segmentation. We run the learning algorithm with a few selected ordinal and nominal logistic models. In all cases we perform 20 runs of the algorithm with different random initializations of the latent variables $\boldsymbol{L}$. The results shown in the following are the segmentations that achieved the highest likelihood score (6) within the 20 restarts.

Figure 4 shows the result of clustering with the (3o,3o) and (3n,3n) logistic models. We use different colors to represents segments computed by nominal logistic models, and greyscale values for ordinal logistic models. The greyscale values then show the ordering of the segments according to their (ordinal) index. One first observes that both models have produced one segmentation in which the mountain areas are identified as one segment: there is an almost perfect correspondence between the mountain attribute illustrated in Figure 1, and the dark grey, respectively yellow, segments in the first segmentations of Figure 4



(a) (b)

Fig. 3. Mutual information: synthetic data (a), real data (b)

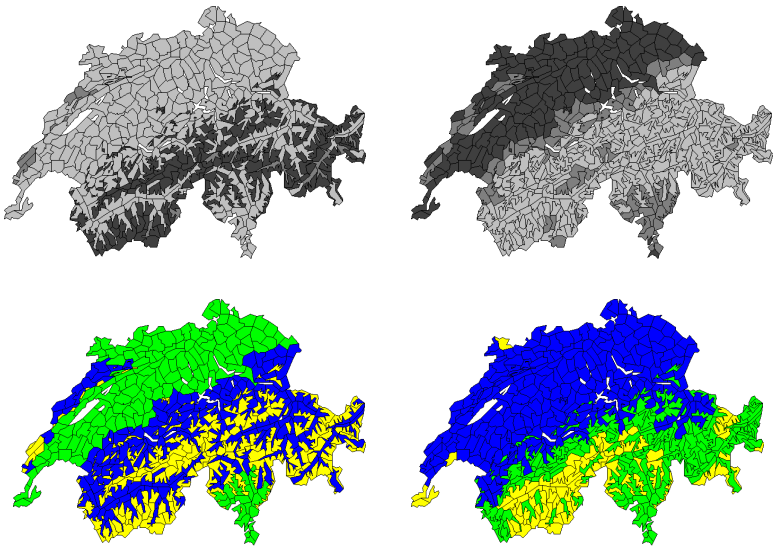


Fig. 4. Clustering result using (3o,3o) (top) and (3n,3n) (bottom) logistic models

(note that our method does not entail an ordering of the different segmentations; in particular, in Figure 4 we have just for convenience vertically aligned similar segmentations, and arbitrarily put the ones containing the mountain segment first).

Apart from the mountain/valley attribute our data does not contain "hidden class variables" that could be used for interpreting the segmentations, and therefore one has to look for additional, external data sources, and expert knowledge. As previously mentioned, we expect that the different segmentations to some extent correspond to ecological factors that determine plant growth. A difficulty we now encounter is that many such candidate factors (e.g., average annual temperature, average precipitation) are highly correlated with the mountain/valley division, and often show a secondary gradient in north-south direction. The second segmentations of Figure 4 are somewhat dominated by a north-south stratification, and also exhibit some of the patterns visible in Figure 1 (b). However, it seems impossible to identify these north-south segmentations with any particular ecological factor. Instead, it can only be taken as aggregating the north-south dependency of several factors. Moreover, whereas one clustering showing the mountain/valley division was quite consistently produced in the random restarts, there were larger variations observed in the north-south clustering.

The range of likelihood values obtained in 20 restarts was $-288 \cdot 10^3$ to $-278 \cdot 10^3$ for (3o,3o) clustering, and $-308 \cdot 10^3$ to $-296 \cdot 10^3$ for (3n,3n) clustering (since the former model fits more independent parameters, higher likelihood scores are to be expected).

Figure 3 shows the mutual information values for the plant features and the (3o,3o) segmentation of Figure 4. This plot shows a relatively strong cor-

relation of some species with the mountain/valley clustering, and a somewhat less pronounced primary correlation of some other species with the north/south segmentation. The large number of species with very small mutual information values for either segmentation is largely made up of species that occur in only a few areas.

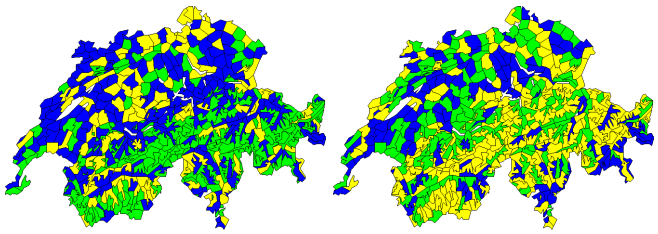


Fig. 5. Clustering result using multinomial $P(\boldsymbol{X} \mid \boldsymbol{L})$

In analogy to the experiment with synthetic data, we also perform with the real data an experiment with full multinomial species distribution models instead of the logistic ones. The result is shown in Figure 5. While the mountain/valley pattern is also partly visible in some of the segments, there is no single segment or segmentation corresponding to this division, and the overall segmentation result is clearly less useful than the one obtained with the logistic models.

The poor performance of the multinomial model may in part be due to this model's inability to isolate in its different clusterings several independent explanatory factors, as illustrated by the synthetic data experiments. In addition, the multinomial model suffered from severe problems of convergence to local optima: even though the global likelihood maximum of the multinomial model must be at least as high as the logistic optimum, the likelihood values found in 20 restarts were significantly lower for the multinomial than for the logistic models (range $-433 \cdot 10^3$ to $-405 \cdot 10^3$).

The average time consumption of a single run (restart) of (3o,3o) or (3n,3n) clustering was approximately 3 hours, with an average of approximately 15 iterations until convergence. This increased to approximately 6 hours for (3o,3o,3o) or (3n,3n,3n) clusterings. The time is consumed almost entirely in fitting in each iteration the 2398 logistic regression models for all the plants. For comparison, a single run with the multinomial model (taking approximately 8 iterations on average until convergence) takes only about 1 minute, since the multinomial model is easily fit by taking simple counts.

6 Discussion and Future Work

Our experiments have shown that using FL-clustering we can find multiple meaningful clusterings of categorical data. The objective in our approach is explana-

tory (identify underlying factors that determine the overall data patterns) rather than descriptive (provide the user with multiple views of the data).

For our purpose, it is clearly essential to use a conditional model $P(\boldsymbol{X} \mid \boldsymbol{L})$ of a restricted functional form, rather than an unconstrained multinomial model. Logistic regression models are a canonical choice, and can be seen as a categorical data analogue to the linear mappings between latent and observed dimensions in the factor analysis model.

A common objective in multiple clustering is that different clusterings are in some sense orthogonal or complementary. We are not yet able to say in which sense, or to what extent, FL-clustering satisfies such an objective. Empirically, a bias towards learning complementary clusterings was difficult to verify with our data, since most natural candidate segmentations based on hidden environmental variables would exhibit rather similar patterns (and not at all resemble the segmentations in Figure 2). Theoretically, one can note that a multi-clustering $\boldsymbol{L} = \boldsymbol{l}$ in which two clusterings are identical can not be a local maximum of the likelihood (6) (except for some degenerate, noise-free, data sets). FL-clustering, thus, is biased away from returning multiple identical clusterings. How this can be strengthened into a formal result linking likelihood gain and complementarity of different clusterings is a subject for future work.

In our experiments we have used data with a spatial structure on the data instances. Within this paper, we have used the spatial structure only for the visualization of the clustering (i.e., segmentation) results. The model can equally be used for other categorical data, and is especially suited for high-dimensional binary data (such as text document data).

On the other hand, our work was also specifically motivated by spatial data, and the relationship in this case of multiple clustering with factorial hidden Markov models [3] and factorial Markov random fields [7]. For spatial data one can impose a Markov random field structure on the latent variables $\boldsymbol{L}$, i.e., the assumption of a uniform distribution for $\boldsymbol{L}$ which we used to derive (6) is replaced, e.g., by the assumption that $P(\boldsymbol{L} = \boldsymbol{l} \mid \theta^*_{\boldsymbol{L}})$ is a Gibbs distribution with fixed parameters $\theta^*_{\boldsymbol{L}}$. Learning in such a setting proceeds in the same way as described in Section 4, only that $P(\boldsymbol{L} = \boldsymbol{l} \mid \theta^*_{\boldsymbol{L}})$ has to be added as a likelihood factor. The optimization in step ii will then usually not be possible precisely, and require an approximate solution. In this paper we did not employ a Markov random field model, since this would usually be used to enforce some smoothness and contiguity properties of the learned segments, which, for our data, seems unwarranted (considering, e.g., the rugged outline of the mountain areas).

In this paper we focused on the core of a probabilistic (multi-) clustering model, i.e., the joint distribution of latent and observable variables. In this model, the number of clusterings, and the number of clusters in each clustering is fixed. We remark, however, that either model selection techniques like BIC or MDL scoring, or a nonparametric Bayesian 'wrapper' around the core model can be used to also learn the model structure.

7 Conclusion

We proposed a latent variable model for multiple clustering of categorical data based on a logistic regression model for the conditional distribution of the observed features. We believe that in analogy to successful techniques for dimensionality reduction, a restricted distributional form for the noisy transformation between the latent and the observed features can be instrumental for revealing relevant patterns in the latent feature space.

For clustering based on a latent variable model we have suggested a simple optimization of the conditional likelihood of the data given the latent variables, with a fixed marginal distribution for the latent variables. This leads to a learning procedure that is linear in the number of observed features, and enables us to experiment with high-dimensional biogeographical data. Our preliminary results from these experiments demonstrate the ability of the method to discover clusterings that represent meaningful explanatory factors for the data. However, further work is needed to consolidate the results returned for this data, and to investigate their potential biological meaning.

References

1. A. Agresti. *Categorical Data Analysis*. Wiley, 2002.
2. T. Chen, N. Zhang, T. Liu, K. M. Poon, and Y. Wang. Model-based multidimensional clustering of categorical data. *Artificial Intelligence*, 2011. To appear.
3. Z. Ghahramani and M. Jordan. Factorial hidden markov models. *Machine Learning*, 29:245–273, 1997.
4. Y. Guan, J. G. Dy, D. Niu, and Z. Ghahramani. Variational inference for nonparametric multiple clustering. In *KDD10 Workshop on Discovering, Summarizing and Using Multiple Clusterings*, 2010.
5. J. A. Hagenaars. *Loglinear Models with Latent Variables*. Number 94 in Quantitative Applications in the Social Sciences. Sage Publications, 1993.
6. P. Jain, R. Meka, and I. Dhillon. Simultaneous unsupervised learning of disparate clusterings. In *SIAM Int. Conf. on Data Mining*, pages 858–869, 2008.
7. J. Kim and R. Zabih. Factorial markov random fields. In *Proc. of ECCV 2002*, number 2352 in LNCS, pages 321–334, 2002.
8. D. Niu, J. G. Dy, and M. I. Jordan. Multiple non-redundant spectral clustering views. In *Proc. of the 27th Int. Conf. on Machine Learning (ICML-10)*, 2010.
9. L. Orloci. An agglomerative method for classification of plant communities. *The Journal of Ecology*, 55(1):193–206, 1967.
10. H. Wang, H. Shan, and A. Banerjee. Bayesian cluster ensembles. *Statistical Analysis and Data Mining*, 4(1):54–70, 2011.
11. T. Wohlgemuth. Biogeographical regionalization of switzerland based on floristic data: How many species are needed? *Biodiversity Letters*, 3(6):180–191, 1996.
12. T. Wohlgemuth. Ein floristischer ansatz zur biogeographischen gliederung der schweiz. *Botanica Helvetica*, 106:227–260, 1996.
13. N. L. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:697–723, 2004.
14. N. L. Zhang, Y. Wang, and T. Chen. Discovery of latent structures: Experience with the COIL challenge 2000 data set. *Journal of Systems Science and Complexity*, 21:172–183, 2008.

Subjectively Interesting Alternative Clusters

Tijl De Bie

Intelligent Systems Laboratory, University of Bristol, UK
tijl.debie@gmail.com
http://www.tijldebie.net

Abstract. We deploy a recently proposed framework for mining subjectively interesting patterns from data [3] to the problem of clustering, where patterns are clusters in the data. This framework outlines how subjective interestingness of patterns (here, clusters) can be quantified using sound information theoretic concepts. We demonstrate how it motivates a new objective function quantifying the interestingness of (a set of) clusters, automatically accounting for a user's prior beliefs and for redundancies between the clusters.
Directly searching for the optimal set of clusters defined in this way is hard. However, the optimization problem can be solved to a provably good approximation if clusters are generated iteratively, paralleling the iterative data mining setting discussed in [3]. In this iterative scheme, each subsequent cluster is maximally interesting given the previously generated ones, automatically trading off interestingness with non-redundancy. Thus, this implementation of the clustering approach can be regarded as a method for alternative clustering. Although generating each cluster in an iterative fashion is computationally hard as well, we develop an approximation technique similar to spectral clustering algorithms.
We end with a few visual demonstrations of the alternative clustering approach to artificial datasets.

Keywords: Subjective interestingness, alternative clustering

1 Introduction

A main challenge in research on clustering methods and theory is that clustering is (in a way intentionally) ill-defined as a task. As a result, numerous types of syntaxes for cluster patterns have been suggested (e.g. clusters as hyperrectangles, hyperspheres, ellipsoids, decision trees; clusterings as partitions, hierarchical partitionings, etc). Additionally, even when the syntax is fixed, there are often various alternative choices for the objective function (e.g. the K-means cost function, the likelihood of a mixture of Gaussians, etc).

Despite this variety in approaches, the goal of clustering is almost always to provide a user with insight in the structure of the data, allowing the user to conceptualize it as coming from a number of broad areas in the data space. The knowledge of such a structure can be more or less elucidating to the user, also depending on the prior beliefs the user held about the data.

Here, we take the perspective that a clustering is more useful if it conveys more novel information. We make a specific choice for a cluster syntax, and we deploy ideas from [3] to quantify the interestingness of a cluster as the amount of information conveyed to the user when told about the cluster's presence.

Our approach attempts to quantify *subjective* interestingness of clusters, in that it takes prior beliefs held by the user into account. As a result, different clusters might be deemed interesting to different users. One particular example is the situation where a user has already been informed about previously discovered clusters in the data, which is the *alternative clustering* setting. In that case, clusters that are individually informative while non-redundant will be the most interesting ones. Our approach naturally deals with alternative clustering, by regarding already communicated clusters as prior beliefs.

Throughout this paper $\mathbf{x} \in \mathbb{R}^d$ denotes a d-dimensional data point, and $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1' \ \mathbf{x}_2' \cdots \mathbf{x}_n' \end{pmatrix}'$ denotes the data matrix containing n data points as its rows. The space the data matrix belongs to is denoted as $\mathcal{X} = \mathbb{R}^{n \times d}$.

2 A framework for data mining: a brief overview

For completeness, we here provide a short overview of a framework for data mining that was introduced in [3], and readers familiar with this paper can skip this section. Earlier and more limited versions of this framework, as well as its application to frequent pattern mining, can be found in [4, 2, 5]. For concreteness, here we specialize the short overview of the framework to the case where the data is a data set, summarized in the data matrix $\mathbf{X}$.

The framework aims to formalize data mining as a process of information exchange between the data and the data miner (the user). The goal of the data miner is to get as good an understanding about the data as possible, i.e. to reduce his uncertainty as much as possible. To be able to do this, the degree of uncertainty must be quantified, and to this end we use probability distribution P (referred to as the *background distribution*) to model the prior beliefs of the user about the data $\mathbf{X}$, in combination with ideas from information theory.

More specifically, the framework deals with the setting where the prior beliefs specify that the background distribution belongs to one of a set $\mathcal{P}$ of possible distributions. The more prior beliefs, the smaller this set will be. For example, the data miner may have a set of prior beliefs that can be formalized in the form of constraints the background distribution P satisfies:

$$\int_{\mathbf{X} \in \mathbb{R}^{n \times d}} f_i(\mathbf{X}) P(\mathbf{X}) = c_i, \ \forall i.$$

Such constraints represent the fact that the expected value of certain statistics (the functions f_i) are equal to a given number. The set $\mathcal{P}$ is defined as the set of distributions satisfying these constraints. (Note that the framework is not limited to such prior beliefs, although they are convenient from a practical viewpoint.)

We argued in [3] that among all distributions $P \in \mathcal{P}$, the 'best' choice for P is the one of maximum entropy given these constraints. This background distribution is the least biased one, thus not introducing any other undue constraints

on the background distribution. A further game-theoretic argument in favour of using the distribution of maximum entropy is given in [3].

In the framework, a pattern is defined as any piece of knowledge about the data that reduces the set of possible values it may take from the original data space $\mathcal{X} = \mathbb{R}^{n \times d}$ to a subset $\mathcal{X}'$. We then argued that the subjective interestingness of such a pattern can be adequately formalized as the *self-information* of the pattern, i.e. the negative logarithm of the probability that the pattern is present in the data, i.e. by $-\log\left(P(\mathbf{X} \in \mathcal{X}')\right)$. Thus, patterns are deemed more interesting if their probability is smaller under the background model, and thus if the user is more surprised by their observation.

After observing a pattern, a user instinctively adapts his beliefs. In [3] we argued that a natural and robust way to model this is by updating the background distribution to a new distribution P' defined as P conditioned on the pattern's presence. The self-information of subsequent patterns can thus be evaluated by referring to the new background distribution P', and so on in an iterative fashion.

In [3] we showed that mining the most informative *set* of patterns formally corresponds to a weighted set coverage problem, attempting to cover as many elements from the set $\mathcal{X}$ that have a small probability under the initial background distribution P. This problem is NP-hard, but it can be approximated well by a greedy approach, and the iterative data mining approach is equivalent to such a greedy approximation.

Thus, the alternative clustering method we will detail below generates clusters in an iterative manner, in such a way that at any time the clusters generated so far are approximately the most informative set of clusters of that size.

3 Subjective interestingness of clusters

3.1 Prior beliefs and the maximum entropy background distribution

Here we consider two types of initial prior beliefs, expressed as constraints on the first and second order cumulants of the data points. It is conceptually easy to extend the results from this paper to other types of prior beliefs, although the computational cost will vary. The background distribution incorporating these constraints is the maximum entropy distribution that has the prescribed first and second order cumulants. It is well known that for data with unbounded domain, this distribution is the multivariate Gaussian distribution with mean and covariance matrix equal to the prescribed cumulants:

$$P(\mathbf{X}) = \frac{1}{\sqrt{(2\pi)^{nd}|\boldsymbol{\Sigma}|^n}} \exp\left(-\frac{1}{2}\text{trace}\left[(\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')'\right]\right). \quad (1)$$

We note that the prescribed cumulants may be computed from the actual data at the request of the data miner, such that they are indeed part of the prior knowledge. However, they may also be beliefs, in the sense that they may be based on external information or assumptions that may be right or wrong.

3.2 A syntax for cluster patterns

The framework from [3] was developed for patterns generally defined as properties of the data. Thus, a pattern's presence in the data constrains the set of possible values the data may have, and in this sense the knowledge of the presence of a pattern reduces the uncertainty about the data and conveys information.

In this paper we restrict our attention to one specific type of cluster pattern. The pattern type we consider is parameterized by a set of indices to the data points and a vector in the data space. The pattern is then the fact that the mean of the data points for these indices is equal to the specified vector.

More formally, let $\mathbf{e}_I \in \mathbb{R}^d$ be defined as an indicator vector containing zeros at positions $i \notin I$ and ones at positions $i \in I$, and let $n_I = |I| = \mathbf{e}_I'\mathbf{e}_I$ denote the number of elements in I. Then, our patterns are constraints of the form:

$$\frac{1}{n_I}\sum_{i\in I}\mathbf{x}_i = \boldsymbol{\mu}_I,$$
$$\Leftrightarrow \mathbf{X}'\frac{\mathbf{e}_I}{\mathbf{e}_I'\mathbf{e}_I} = \boldsymbol{\mu}_I.$$

Such a constraint restricts the possible values of the data set $\mathbf{X}$, in that the mean of a subset of the data points is required to have a prescribed value $\boldsymbol{\mu}_I$.

3.3 The self-information of a cluster pattern

The following theorem shows how to assess the self-information of a pattern.

Theorem 1. *Given a background distribution of the form in Eq. (1), the probability of a pattern of the form* $\mathbf{X}'\frac{\mathbf{e}_I}{\mathbf{e}_I'\mathbf{e}_I} = \boldsymbol{\mu}_I$ *is given by:*

$$P\left(\mathbf{X}'\frac{\mathbf{e}_I}{\mathbf{e}_I'\mathbf{e}_I} = \boldsymbol{\mu}_I\right) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2|I|}\mathbf{e}_I'\cdot\left[(\mathbf{X}-\mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X}-\mathbf{e}\boldsymbol{\mu}')'\right]\cdot\mathbf{e}_I\right).$$

Thus the self-information of the pattern for a cluster specified by the set I, defined as the negative log probability of the cluster pattern and denoted as $SelfInformation_I$, is equal to:

$$SelfInformation_I = \frac{1}{2}\log\left((2\pi)^d|\boldsymbol{\Sigma}|\right) + \frac{1}{2}Q_I,$$
$$\text{where } Q_I = \frac{1}{|I|}\mathbf{e}_I'\cdot\left[(\mathbf{X}-\mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X}-\mathbf{e}\boldsymbol{\mu}')'\right]\cdot\mathbf{e}_I.$$

Note that the self-information depends on I only through Q_I, so we may choose to use Q_I as a quality metric for a cluster, as we will do in this paper.

This theorem can be used to quantify the self-information of any cluster given the background model based on the prior beliefs of the data miner. Note however that it cannot be used to assess the self-information of a cluster after other clusters have already been found, as each new cluster will affect the user's prior beliefs. How this can be accounted for will be discussed in Sec. 3.4, based on a generalization of Theorem 1. As Theorem 1 directly follows from Theorem 2, we will only provide a proof for the latter in Sec. 3.4.

3.4 The self-information of a set of cluster patterns

Let us discuss the more general case, where the patterns are constraints of the form $\mathbf{X}'\mathbf{E} = \mathbf{M}$ with $\mathbf{E} \in \mathbb{R}^{n\times k}$ and $\mathbf{M} \in \mathbb{R}^{d\times k}$. Clearly, the type of patterns we wish to consider are a special case of this, with $\mathbf{E} = \frac{\mathbf{e}_I}{\mathbf{e}_I'\mathbf{e}_I}$ and $\mathbf{M} = \boldsymbol{\mu}_I$. Furthermore, it allows us to consider a *composite pattern*, a pattern defined as the union of a set of k patterns. Indeed, if we have k different cluster patterns specified by the sets from $\mathcal{I} = \{I_i\}$, we can write down this set of constraints concisely as $\mathbf{X}'\mathbf{E} = \mathbf{M}$ where $\mathbf{E}$ and $\mathbf{M}$ contain $\frac{\mathbf{e}_{I_i}}{\mathbf{e}_{I_i}'\mathbf{e}_{I_i}}$ and the mean vector $\boldsymbol{\mu}_i$ of the i'th cluster as their i'th columns.

Theorem 2. *Let the columns of the matrix* $\mathbf{E}$ *be the indicator vectors of the sets in* $\mathcal{I} = \{I_i\}$*, and let* $\mathbf{P}_\mathbf{E} = \mathbf{E}(\mathbf{E}'\mathbf{E})^{-1}\mathbf{E}'$*, the projection matrix onto the column space of* $\mathbf{E}$*. Then, the probability of the composite pattern* $\mathbf{X}'\mathbf{E} = \mathbf{M}$ *is given by:*

$$P(\mathbf{X}'\mathbf{E} = \mathbf{M}) = \frac{1}{\sqrt{(2\pi)^{kd}|\boldsymbol{\Sigma}|^k}} \exp\left(-\frac{1}{2} trace\left[\mathbf{P}_\mathbf{E} \cdot (\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')'\right]\right).$$

Thus the self-information of the set of patterns defined by the columns of $\mathbf{E}$*, defined as its negative log probability and denoted as* $SelfInformation_\mathcal{I}$*, is equal to:*

$$SelfInformation_\mathcal{I} = \frac{k}{2}\log\left((2\pi)^d|\boldsymbol{\Sigma}|\right) + \frac{1}{2}Q_\mathcal{I},$$
$$where\ Q_\mathcal{I} = trace\left[\mathbf{P}_\mathbf{E} \cdot (\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')'\right].$$

Again, since the self-information depends on I only through Q_I, we choose to use Q_I as a quality metric for a cluster further below.

Proof. A constraint $\mathbf{X}'\mathbf{E} = \mathbf{M}$ constrains the data $\mathbf{X}$ to an $(n-k)\times d$ dimensional affine subspace in the following way. Let us write the singular value decomposition for $\mathbf{E}$ as:

$$\mathbf{E} = \begin{pmatrix}\mathbf{U}\ \mathbf{U}_0\end{pmatrix}\begin{pmatrix}\boldsymbol{\Lambda} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{V}\ \mathbf{V}_0\end{pmatrix}'.$$

Then, this constraint can be written in the following form:

$$\mathbf{X} = \mathbf{U}\mathbf{Z} + \mathbf{U}_0\mathbf{Z}_0,$$

where $\mathbf{Z} = \boldsymbol{\Lambda}^{-1}\mathbf{V}'\mathbf{M}'$ is a constant fixed by $\mathbf{E}$ and $\mathbf{M}$, and $\mathbf{Z}_0 \in \mathbb{R}^{(n-k)\times d}$ is a variable. In general, writing $\mathbf{X} = \mathbf{U}\mathbf{Z} + \mathbf{U}_0\mathbf{Z}_0$, we can write the probability density for $\mathbf{X}$ as:

$$\begin{aligned}
P(\mathbf{X}) &= P(\mathbf{Z}, \mathbf{Z}_0),\\
&= \frac{1}{\sqrt{(2\pi)^{nd}|\boldsymbol{\Sigma}|^n}} \exp\left(-\frac{1}{2}\text{trace}\left[(\mathbf{U}\mathbf{Z} + \mathbf{U}_0\mathbf{Z}_0 - \mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{U}\mathbf{Z} + \mathbf{U}_0\mathbf{Z}_0 - \mathbf{e}\boldsymbol{\mu}')'\right]\right),\\
&= \frac{1}{\sqrt{(2\pi)^{(n-k)d}|\boldsymbol{\Sigma}|^{n-k}}} \exp\left(-\frac{1}{2}\text{trace}\left[(\mathbf{Z}_0 - \mathbf{U}_0'\mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{Z}_0 - \mathbf{U}_0'\mathbf{e}\boldsymbol{\mu}')'\right]\right)\\
&\quad \cdot \frac{1}{\sqrt{(2\pi)^{(k)d}|\boldsymbol{\Sigma}|^k}} \exp\left(-\frac{1}{2}\text{trace}\left[(\mathbf{Z} - \mathbf{U}'\mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{Z} - \mathbf{U}'\mathbf{e}\boldsymbol{\mu}')'\right]\right)
\end{aligned}$$

We can now compute the marginal probability density for $\mathbf{Z}$ by integrating over $\mathbf{Z}_0$, yielding:

$$P(\mathbf{Z}) = \frac{1}{\sqrt{(2\pi)^{kd}|\boldsymbol{\Sigma}|^k}} \exp\left(-\frac{1}{2}\text{trace}\left[(\mathbf{Z}-\mathbf{U}'\mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{Z}-\mathbf{U}'\mathbf{e}\boldsymbol{\mu}')'\right]\right).$$

The probability density value for the pattern's presence, i.e. for $\mathbf{X}'\mathbf{E} = \mathbf{M}$ or equivalently $\mathbf{Z} = \boldsymbol{\Lambda}^{-1}\mathbf{V}'\mathbf{M}'$, is thus:

$$\begin{aligned}
&P(\mathbf{Z} = \boldsymbol{\Lambda}^{-1}\mathbf{V}'\mathbf{M}') \\
&= \frac{1}{\sqrt{(2\pi)^{kd}|\boldsymbol{\Sigma}|^k}} \exp\left(-\frac{1}{2}\text{trace}\left[(\boldsymbol{\Lambda}^{-1}\mathbf{V}'\mathbf{M}' - \mathbf{U}'\mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\boldsymbol{\Lambda}^{-1}\mathbf{V}'\mathbf{M}' - \mathbf{U}'\mathbf{e}\boldsymbol{\mu}')'\right]\right), \\
&= \frac{1}{\sqrt{(2\pi)^{kd}|\boldsymbol{\Sigma}|^k}} \exp\left(-\frac{1}{2}\text{trace}\left[\mathbf{P}_\mathbf{E}\cdot(\mathbf{X}-\mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X}-\mathbf{e}\boldsymbol{\mu}')'\right]\right),
\end{aligned}$$

where $\mathbf{P}_\mathbf{E} = \mathbf{E}(\mathbf{E}'\mathbf{E})^{-1}\mathbf{E}'$ is a projection matrix projecting onto the k-dimensional column space of $\mathbf{E}$. □

Note that Theorem 1 is indeed a special case of Theorem 2 as can be seen by substituting $\mathbf{E} = \mathbf{e}_I$ and $\mathbf{P}_\mathbf{E} = \frac{\mathbf{e}_I\mathbf{e}_I'}{|I|}$.

According to the framework, a (composite) pattern specified by matrices $\mathbf{E}$ and $\mathbf{M}$ in this way is thus more informative if trace $\left[\mathbf{P}_\mathbf{E}\cdot(\mathbf{X}-\mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X}-\mathbf{e}\boldsymbol{\mu}')'\right]$ is larger. Thus, we could search for the most informative *set* of clusters by maximizing this quality measure with respect to a set $\mathcal{I} = \{I_i\}$ of clusters. This is a hard problem though, even in the case where only one cluster is sought. Thus, we developed an approximation algorithm.

Our approach is approximate in two ways. First, the search for a set of clusters is approximated using a greedy algorithm, searching clusters one by one, thus operating like an alternative clustering algoritm. From [3], it can be seen that this greedy approximation is guaranteed to approximate the true optimum provably well. Second, the search for each cluster is relaxed to an eigenvalue problem. These issues are discussed in greater detail in Sec. 4.

3.5 The cost of describing a cluster

The framework from [3] suggests to take into account not only the self-information of a pattern, but also the cost to communicate a pattern, i.e. its description length. This depends on the coding scheme used, which should reflect the perceived complexity of a pattern as perceived by the data miner. Choosing this coding scheme can also be done so as to bias the results toward specific patterns.

In the current context, describing a pattern amounts to describing the subset I and the mean vector $\boldsymbol{\mu}_I$. For simplicity, we assume the description length is constant for all patterns, independent of I and $\boldsymbol{\mu}_I$. However, note that different costs could be used if patterns with smaller sets I are more easy to understand (i.e. have a smaller cost), or vice versa.

4 Alternative clustering: finding the next most informative cluster

Here we discuss an iterative approach to optimizing the quality measure $Q_{\mathcal{I}}$ from Theorem 2. There are two reasons for choosing an iterative approach.

Firstly, directly optimizing the quality measure is equivalent to a set covering type problem (see [3] for more background on why this is the case). While NP-hard, this problem can be approximated well by optimizing over the different clusters (and thus the columns of $\mathbf{E}$) in a greedy iterative manner.

Secondly, usually it is not a priori clear how many clusters are required for the data miner to be sufficiently satisfied with his new understanding of the data. The idea of alternative clustering, as we view it, is to provide the user the opportunity to request new clusters (or clusterings) as long as more are desired. Optimizing the quality measure over a growing set of clusters by iteratively optimizing over a newly added column of $\mathbf{E}$ is thus a type of alternative clustering.

Hence, the iterative approach can be regarded as an approximation, but one with usability benefits over a global optimizing approach.

4.1 The iterative scheme: alternative clustering

To search for the first cluster, we attempt to optimize the quality function from Theorem 1. This is itself a hard problem, but we explain how we approximately solve it in Sec. 4.2.

In the subsequent iterations, let us say that we have already found $k-1 \geq 1$ clusters, and the matrices $\mathbf{E}$ and $\mathbf{M}$ respectively contain the normalized indicator vectors and cluster means as their columns. We are interested in finding the k'th cluster so as to optimize the quality measure from Theorem 2 but keeping the first $k-1$ cluster patterns as they are.

To do this, it is convenient to write the quality measure as a function of the k'th cluster with indicator vector $\mathbf{e}_k$. Let $\mathbf{Q}_{\mathbf{E}} = \mathbf{I} - \mathbf{P}_{\mathbf{E}}$, the projection matrix on the null column space of $\mathbf{E}$. Furthermore, let us denote $\mathbf{E}^* = \left(\mathbf{E}\ \mathbf{e}_k\right)$. Then, using the definition of a projection matrix and the matrix inversion lemma:

$$\begin{aligned}
Q_{\{I_i|i=1:k\}} &= \text{trace}\left[\mathbf{P}_{\mathbf{E}^*} \cdot (\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')'\right], \\
&= \text{trace}\left[\mathbf{P}_{\mathbf{E}} \cdot (\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')'\right] \\
&\quad + \text{trace}\left[\frac{\mathbf{Q}_{\mathbf{E}}\mathbf{e}_k\mathbf{e}_k'\mathbf{Q}_{\mathbf{E}}}{\mathbf{e}_k'\mathbf{Q}_{\mathbf{E}}\mathbf{e}_k} \cdot (\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')'\right], \\
&= Q_{\{I_i|i=1:k-1\}} + \frac{\mathbf{e}_k'\left[\mathbf{Q}_{\mathbf{E}} \cdot (\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')'\mathbf{Q}_{\mathbf{E}}\right]\mathbf{e}_k}{\mathbf{e}_k'\mathbf{Q}_{\mathbf{E}}\mathbf{e}_k}.
\end{aligned}$$

Note that if we define $Q_{\emptyset} = 0$ and with $\mathbf{Q}_{\mathbf{E}} = \mathbf{I}$ the quality measure from Theorem 1 is retrieved. We can thus interpret the above reformulation of the quality metric for the k'th cluster conditioned on the first $k-1$ clusters as being the quality metric for a first cluster on data that is projected onto the space

orthogonal to the $k-1$ columns of $\mathbf{E}$, i.e. the $k-1$ previously selected indicator vectors. It is as if the data was deflated to take account of the knowledge of the previously found cluster patterns, thus automatically accounting for redundancy.

4.2 A spectral relaxation of the iterations

Each of the iterative steps thus reduces to the maximization of the following increase of the quality measure:

$$\Delta Q_k = \frac{\mathbf{e}_k' \left[\mathbf{Q_E} \cdot (\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')'\mathbf{Q_E}\right] \mathbf{e}_k}{\mathbf{e}_k'\mathbf{Q_E}\mathbf{e}_k}. \tag{2}$$

If we relax the vector $\mathbf{e}_k$ to be real-valued instead of containing only 0's and 1's, this Rayleigh quotient is maximized by the dominant eigenvector of the matrix $\mathbf{Q_E} \cdot (\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')\boldsymbol{\Sigma}^{-1}(\mathbf{X} - \mathbf{e}\boldsymbol{\mu}')'\mathbf{Q_E}$. Thus, as an approximation technique we will use this dominant eigenvector, and threshold it to obtain a crisp 0/1 vector $\mathbf{e}_k$. To determine a suitable threshold, we simply do an exhaustive search over $n+1$ threshold values that generate a different set I of indices $i \in I$ for which $\mathbf{e}_k(i) = 1$, selecting the threshold that maximizes the quantity in Eq. (2).

4.3 A kernel-based version

Note that for $\boldsymbol{\Sigma} = \mathbf{I}$, the quality metrics depend on $\mathbf{X}$ only through the inner product matrix $\mathbf{X}\mathbf{X}'$. This means that a kernel-variant is readily derived, by substituting this inner product matrix with any suitable kernel matrix. In this way nonlinearly shaped clusters can be obtained, similar to spectral clustering methods and kernel K-Means.

5 Relations to existing work

There appear to be strong relations between spectral clustering and our spectral relaxation of the problem [7]. Additionally, the quality measure is strongly related to the K-Means cost function [8, 6]. Finally, there seem to be interesting connections to (0-1) SDP problems used for solving combinatorial optimization problems such as clustering (e.g. K-Means and graph cut clustering) [6, 1].

6 Experiments

We conducted 3 experiments, each time reporting the result of 6 iterations of the alternative clustering scheme. Initial prior beliefs are always $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma} = \mathbf{I}$.

- A plain application to a synthetic dataset of 100 points and 2 dimensions in 4 clusters. See Fig. 1.
- An application to the same data but now with a Radial Basis Function (RBF) kernel used for the inner products. See Fig. 2.
- An application with an RBF kernel to a different synthetic dataset, with one central cluster and two half-moon shaped clusters around this central cluster. See Fig. 3.

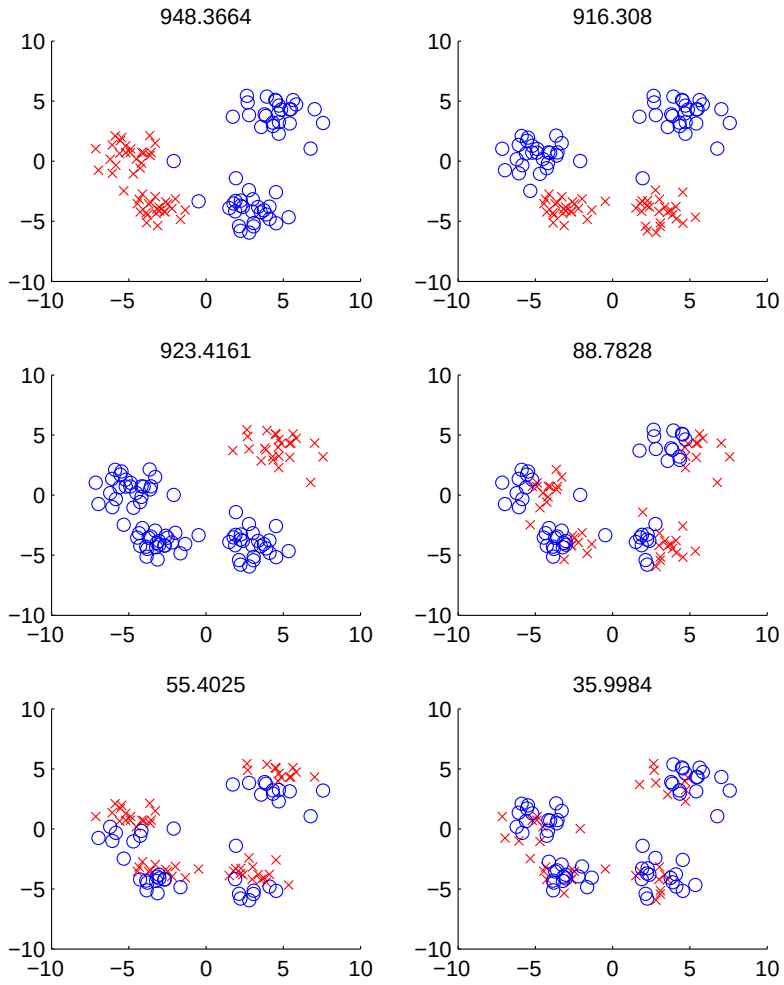


Fig. 1. A synthetic dataset with 25 data points sampled from each of 4 2-dimensional Gaussian distributions with identity covariance matrix and different means. From left to right and top to bottom, the plots show the first 6 consecutive alternative clusters found by our method when a standard inner product is used (data points belonging to the cluster are plotted using crosses). The numbers above the plots are the values of ΔQ_k from Eq. (2) for the cluster shown. Note that it is high for the first 3 clusters, which reveal the enforced cluster structure, before dropping to a much lower level.

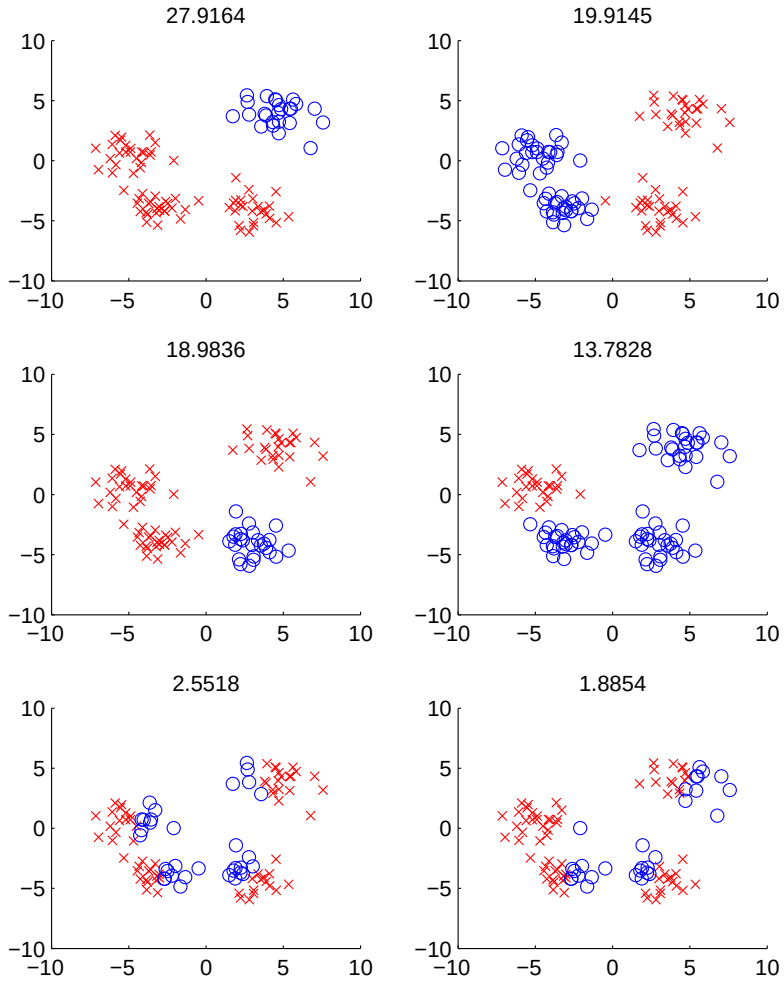


Fig. 2. A synthetic dataset with 25 data points sampled from each of 4 2-dimensional Gaussian distributions with identity covariance matrix and different means. From left to right and top to bottom, the plots show the first 6 consecutive alternative clusters when an RBF kernel with kernel width 3 is used. The numbers above the plots are the values of ΔQ_k from Eq. (2) for the cluster shown.

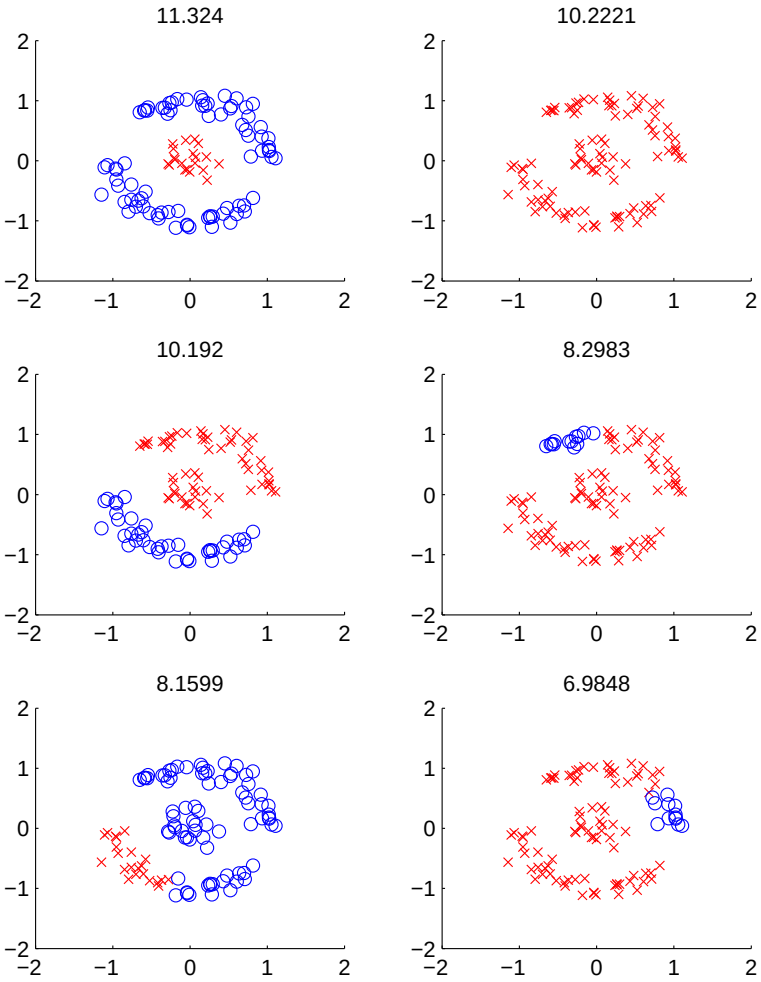


Fig. 3. A synthetic dataset with a central set of 20 data points surrounded by two half moons of 40 data points each. The plots show the first 6 consecutive alternative clusters when an RBF kernel with kernel width 0.3 is used. The numbers above the plots are the values of ΔQ_k from Eq. (2) for the cluster shown. Note that the second cluster generated contains all data points. This is possible and sensible from the perspective of our approach if the mean of the entire data set is significantly different from the expected mean in the initial background model. This may well be the case when working in a Hilbert space induced by the RBF kernel, where all data points lie in one orthant such that their mean cannot be in the origin.

7 Conclusions

In [3] we introduced a framework for data mining, aiming to quantify the subjective interestingness of patterns. We showed that Principal Component Analysis can be seen as implementing this framework for a particular pattern type and prior beliefs, thus providing an alternative justification for this method. In earlier work we also showed the potential of the framework in quantifying subjective interestingness for frequent itemset mining [2, 4, 5]. Now, in the present paper, we showed in detail how the framework can also be applied successfully to the case of clustering, leading to a new approach for alternative clustering that presents subjectively interesting clusters in data in an iterative data mining scheme.

In further work, we will investigate the quality of the spectral relaxation, and consider the development of tighter relaxations (e.g. to semi-definite programs). We will also further develop links with spectral clustering and other existing clustering approaches, to provide alternative justifications and insights or to improve on these approaches. We will also investigate the use of other pattern syntaxes for cluster(ing)s, and the use of more complex types of prior beliefs. Lastly, we plan to demonstrate the power of the framework by also applying these ideas to other types of data and corresponding types of prior beliefs, such as positive real-valued, integer, binary, and more structured types of data.

Due to space constraints, in this workshop paper we could not situate the contributions within the wider literature on alternative clustering. We will rectify this important shortcoming in a later version of this workshop paper.

Acknowledgements
This work is supported by the EPSRC grant EP/G056447/1.

References

1. T. De Bie and N. Cristianini. Fast sdp relaxations of graph cut clustering, transduction, and other combinatorial problems. *Journal of Machine Learning Research*, 7:1409–1436, 2006.
2. T. De Bie. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. Data Mining and Knowledge Discovery, 2010.
3. T. De Bie. An information-theoretic framework for data mining. In *Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD11)*, 2011.
4. T. De Bie, K.-N. Kontonasios, and E. Spyropoulou. A framework for mining interesting pattern sets. *SIGKDD Explorations*, 2010.
5. K.-N. Kontonasios and T. De Bie. An information-theoretic approach to finding informative noisy tiles in binary databases. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, 2010.
6. Jiming Peng and Yu Wei. Approximating k-means-type clustering via semidefinite programming. *SIAM Journal on Optimization*, 18(1), 2007.
7. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
8. H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for k-means clustering. In *Advances in Neural Information Processing Systems 14 (NIPS01)*, pages 1057–1064, 2002.

Evaluation of Multiple Clustering Solutions

Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek

Ludwig-Maximilians-Universität München
Oettingenstr. 67, 80538 München, Germany
http://www.dbs.ifi.lmu.de
{kriegel,schube,zimek}@dbs.ifi.lmu.de

Abstract. Though numerous new clustering algorithms are proposed every year, the fundamental question of the proper way to evaluate new clustering algorithms has not been satisfactorily answered. Common procedures of evaluating a clustering result have several drawbacks. Here, we propose a system that could represent a step forward in addressing open issues (though not resolving all open issues) by bridging the gap between an automatic evaluation using mathematical models or known class labels and the actual human researcher. We introduce an interactive evaluation method where clusters are first rated by the system with respect to their similarity to known results and where "new" results are fed back to the human researcher for inspection. The researcher can then validate and refine these results and re-add them back into the system to improve the evaluation result.

1 Introduction

A major challenge in the development of clustering algorithms is the proper and useful evaluation. In most cases, a clustering algorithm is evaluated using (i) some internal evaluation measure like cohesion, separation, or the silhouette-coefficient (addressing both, cohesion and separation), (ii) some external evaluation measure like accuracy, precision, or recall w.r.t. some given class-structure of the data. In some cases, where evaluation based on class labels does not seem viable, (iii) careful (manual) inspection of clusters shows them to be a somehow meaningful collection of apparently somehow related objects.

All these approaches certainly have their merits but also serious drawbacks.

(i) The evaluation w.r.t. some internal evaluation measure does nothing more than evaluate how well the objective function of the clustering algorithm fits to the chosen evaluation measure. For example, using some compactness measure would be obviously inappropriate to evaluate the results of some density-based clustering [1], simply because density-based clustering does not aim at finding convex clusters. As a consequence, the evaluation does not primarily show that the clustering is meaningful and fitting for the given data. Clusters attributed with good grades could be trivial or rather uninteresting.

(ii) The fundamental problem in using class-labels for evaluation of clustering is the different structure of classes and clusters. Consider for example one

of the best known classification data sets, Fisher's Iris data set [2]. It comprises four descriptors of the Iris flower, namely length and width of petals and sepals, respectively. These descriptors are collected for individual flowers of three different species. The classes are well defined (though not trivial to learn) by some separating borders between members of the classes. The natural clusters in this data set, however, are certainly not evolved according to such (predefined?) borders. Cluster analysis of these data would discover that *I. setosa* is much more different from both, *I. versicolor* and *I. virginica*, than these two are from each other (in fact, they will usually be considered a single cluster). Accordingly, most classification algorithms set out with learning some separating borders between different classes. Opposed to that, clustering algorithms aim at grouping similar objects together. As a consequence, the evaluation of new clustering algorithms towards learning a class structure may introduce some strong bias in the wrong direction into the development and design of new clustering algorithms. Actually, it could be a good and desirable result if a clustering algorithm detects structures considerably different from previously known classes. In that case, the clustering algorithm should not be punished by using some evaluation measure biased towards rediscovery of classes. A more thorough discussion of this issue, along with many more examples, has been provided in [3].

(iii) The third approach, (manual) inspection of clusters and reviewing them for prevalent representation of some meaningful concept, could be figured as 'evaluation by example'. There are attempts to formalize this as 'enrichment' w.r.t. some known concept (this technique is automated to a certain extent in biological analysis of gene data, e.g. [4–10]). In the context of multiple clusterings and overlapping clusters (as are expected in gene data – see the Gene Ontology [11] –, but also in many benchmark data sets that sparked interest of researchers in alternative or multiview clustering, e.g. [12–17], see also [18]) it becomes even more important to find methods of evaluating clusterings w.r.t. each other, w.r.t. existing knowledge, and w.r.t. their usefulness as interpreted by a human researcher. Though the problem of overlapping ground truths (and, hence, the impossibility of using a flat set of class labels directly) is pre-eminent in such research areas as subspace clustering [19], alternative clustering [16], or multiview clustering [13], it is, in our opinion, actually relevant for all non-naïve approaches to clustering that set out to learn something new and interesting about the world (where 'naïve' approaches would require the world to be simple and the truth to be one single flat set of propositions only).

It is our impression, that the third approach is pointing in the right direction since it tries to assess whether some clustering algorithm actually found some new, valid, and previously unknown knowledge (which is, after all, the whole point in performing data mining [20]). As 'evaluation by example', however, it has never been convincingly impartial and always remained tasting somehow subjective and incomplete. The discussion of evaluation scenarios in [3] pointed out some requirements in an automation of evaluation based on multiple (and possibly some unknown) ground truths. Thus we try to establish some first steps in automation of such a process and to set up an evaluation system to address

at least some of the identified requirements. We see this only as some first steps, the system relies on participation of the community to further advance.

In the following, we describe the preliminary system and the envisioned possibilities of future enhancements (Section 2). Based on the available system, we discuss an illustrative example benchmark data set as a case study (Section 3). We conclude the paper in Section 4.

2 A Clustering Evaluation System

Since a main goal of cluster analysis is the discovery of new and previously unknown knowledge, our evaluation concept is built around the comparison of results to known structure in the data. But instead of just computing a score of how well the clustering resembles a known label structure, we actually try to detect situations where it *deviates* from the known structure. Another key difference is that we not only include the target classes, but essentially include any structure information that we can obtain for the data set.

A key source of information are features of any kind. In order to process complex data such as image or video data, feature extraction is essential and a whole research area of its own. But when working in a cluster analysis context, we should treat the features as known properties of the data, and instead evaluate how much *additional* information the clustering algorithm is able to extract from the data that goes beyond the data already extracted using the feature extraction methods: in particular, when feature extraction itself is already very good, almost any clustering algorithm will appear to perform well, but the performance is essentially increased to not more than a naïve statistic on the features.

2.1 Assisted Evaluation

The general process of an assisted evaluation is an iterative interaction between the computer system and the researcher. The system uses the available information to find feature descriptions of the clusters. Clusters that can not be explained sufficiently well using the existing knowledge are then given to the researcher for further external analysis. Knowledge obtained in this process is then added back into the system as additional features, resulting in better explanations for some clusters and thus in new candidates to be analyzed by the human researcher. When assigning a "usefulness" to the different information fed into the system – for example, a simple color feature will not be considered particularly useful but preexisting knowledge – this can also be used to qualitatively rate the output of an algorithm by the usefulness of the information it was able to discover in the data. Both a supervised or semi-supervised evaluation is here possible. For example, the system could perform an initial analysis of the data set, present these results to the analyst, who can then choose results for a more expensive refinement, manually choose complex feature combinations or refine the parameters of found explanations.

2.2 Challenges

In the task of analyzing the characteristics of a cluster, various challenges arise. For example, the cluster size can vary from micro clusters to clusters that span almost the complete data set, resulting in varying imbalance problems. When searching complex explanations involving e.g. the combination of features, or the intersection or union of known classes, the search space is extremely large and an exhaustive search quickly becomes infeasible. Even comparing a cluster with a single numerical feature is non-trivial. Such a feature will give you an ordering (or scoring) of the objects, but the clusters can still occur anywhere within this scoring. Therefore, we need a general way to measure how relevant the information of a feature (or combination of features) is with respect to a particular cluster. Combining such scorings could be based on any linear or non-linear combination of their scores. The scorings however can be strongly correlated, so that in the end, finding the optimal combination offers little benefit to the analyst.

There are many kinds of features, and we will work with two of the most common types of features in the following: class features that differentiate a particular group of objects from the remainder and numerically scoring and ranking features such as the average brightness of an image. Other types such as "bag of words" can probably be handled well enough by breaking them down into individual scoring features.

2.3 Comparing with Existing Classes

Comparing two clusters has of course been extensively studied, and various measures have been developed (see, e.g., [21]). Much of this research (such as pair counting measures) however is designed to compare two complete partitionings of the data (containing more than one cluster each). In our setting, we are again evaluating single clusters with respect to overlapping classes and scorings. Comparing two clusters however is done using simple measures such as precision, recall, or the F-measure (which represents the harmonic mean of these two):

$$F_1 := \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

A nice property of the F-measure is that both trivial solutions (the empty set and the complete data set) score fairly low due to the product in the numerator. Only when both precision and recall are high at the same time, the F-score will be good. When precision equals recall, they will also be equal to the F-score.

We also use this measure in evaluation of a cluster with respect to a scoring, essentially treating these two cases the same, which we will explain next.

2.4 Comparing Clusters with Scorings

A common way of comparing a two-class problem with a scoring is the evaluation using ROC curves. Instead of evaluating a ranking with respect to a class, we

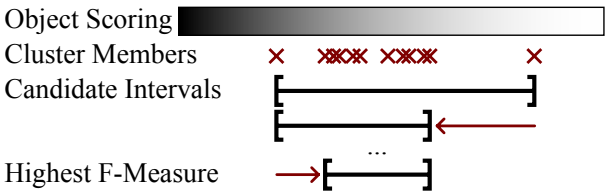


Fig. 1. Evaluating a cluster using the highest F-measure on an interval

could apply ROC curves to evaluate the cluster with respect to the ranking given by the scores. However in our experiments, the results were not very useful: given that the clusters are usually computed on features similar or identical to the reference scorings, a strong correlation and thus a high ROC AUC score between them can be expected. Additionally, ROC is only meaningful when the cluster is at the top or bottom of the scoring, which we would yet have to generalize to allow it to occur at arbitrary positions.

Instead, we chose an approach based on a kind of compactness based on the comparison with classes as discussed before: We search for an arbitrary interval within the scoring that has a high F-measure. Too large an interval will score badly because of a bad precision, while too narrow an interval will suffer from a bad recall. A compact interval containing mostly cluster members however will achieve a high F-measure. In this context, precision can be considered as the density of the cluster members in the interval, while recall is the coverage. Note that this measure is independent of the actual position of the interval within the scoring or the order within the interval. Since we are interested in the *potential* of agreement between the scoring and the cluster, we want to use the maximum F-measure possible; however naïvely there are $O(n^2)$ possible intervals to test. Luckily, we can exploit some monotonicity properties here. Recall obviously is monotonously decreasing, so any subinterval will have at most the same recall. Interesting intervals are thus on the skyline of precision and recall. Intervals which do not have a cluster member on the interval boundary are obviously dominated by the subinterval that fulfills this property (same recall, but better precision). This reduces the search space to $O(k^2)$ for cluster size k. However, we perform a greedy search by starting with the smallest interval containing all cluster members (so at recall 1), then repeatedly narrow down the interval as sketched in Figure 1 by trying to cut off leading and trailing cluster members along with any non-member as long as we can improve the F-measure this way by improving precision at the cost of recall in at most k iterations.

Ties need special handling: an interval may never split within a tie. Then we can map an existing class to a scoring by setting all members to 1 and non-members to 0. If there is some overlap between the test cluster and the known class, the result will be the F-score.

2.5 Scoring Combinations

In addition, we perform a greedy search for a simple additive combination of features. In a preprocessing step, we normalized the scores of each scoring to unit variance to improve results in this step. In the greedy combination phase, we now combine the top matching results by just adding their scores and testing the new scoring. When the combined scoring performs better by a sufficiently large amount, we add it to the candidate list. While we only test a very simple combination of features – not even considering full linear combinations – this greedy search was very successful in our experiments in finding better explanations than single features. We will show examples of this in the next section. But obviously there is much room for improved heuristics in finding such combined explanations.

2.6 Result Presentation

There are essentially infinitely many combinations possible, and even when just using the additive combinations we have theoretically $O(2^r)$ scores for each cluster. The top score itself is often not very useful to the analyst: it may be just one of many very similar explanations. The most interesting analysis results occur when a combination of scorings offers a significantly better explanation than the individual single features, or when there was not found any adequate explanation at all. Therefore we need to make a selection of the results to present to the user. As a heuristic, we will present a result to the user if it is the best single-feature explanation or if no other score with a single feature added or removed performed better. Additionally, we will stop once a threshold of matches has been reached by the accumulated amount explained. Other application-domain specific heuristics may be useful, for example when there is a large number of correlated features.

3 A Case Study

For the case study, we started to analyse the ALOI [22] image data set. It consists of 110250 images of 1000 objects taken from 72 angles and in a series of controlled light conditions varying both color temperature and lighting angle. This metadata can be used to obtain a couple of overlapping classes on the data set, resembling the object number, the viewing angle, the lighting angle, lighting color temperature, and a stereo image shift. Some of these classes are however only useful for machine learning tests; in particular the rotation and stereo image shift usually require a training set and optimized color features.

In addition to these labels we compute some simple color analysis on the pictures. We defined a set of 77 colors spaced evenly in HSV color space (18 hues with 100% and 50% each in saturation and brightness plus 5 grey values for saturation 0%), then computed the average pixel color similarity to these colors for each image to obtain object reference scorings.

For the actual algorithm, we independently produced traditional color histograms in HSV color space with 28 dimensions: 7 bins in hue and 2 bins in saturation and brightness each. In contrast to the features above, the histogram dimensions are not independent, but each pixel is assigned to the closest histogram bin only, so the histograms add up to 1. While the performance of the histograms is of course expected to be similar to the other color features, we wanted to avoid using identical features to not overfit our analysis method.

Early analysis on the objects in this data set allowed us to identify various groups of objects that form sensible clusters aggregating multiple objects such as different jam cans. These additional human-verified clusters sometimes form a hierarchy: for example there are multiple yellow rubber ducks that can be considered a cluster, but there also is a red rubber duck that can be added to form an "any-color rubber duck" cluster. However, there were also some interesting additional features hidden in the data set that were surprisingly useful in explaining results. We highlight these features using a bold typeface and we will explain these features in the discussion below.

We ran OPTICS [23] on the 28 dimensional HSV histograms using Manhattan distance (since this is a rather large data set, and we can use an R^*-tree [24] for acceleration here; on normalized vectors, Manhattan distance equals histogram intersection distance [25]; all implementations featured by ELKI [26]). We chose minPts = 15, $\epsilon = 0.3$ (solely for performance improvements) and $\xi = 0.03$ and obtained a hierarchy of 1442 clusters. The median size is 40 objects, the largest cluster contains 343 images. OPTICS is not a subspace clustering algorithm, but it is a truly hierarchical clustering algorithm, so certain types of overlap among clusters occur. While the majority of objects was not clustered using these unoptimized parameters, the detected clusters were still interesting to analyze. We will give some examples here.

There is a cluster that contains 18 images from object 938 and 19 images from object 939. Some sample images are shown in Figure 2. The cluster is not very surprising, as the two objects are indeed very similar – considering the back side of the objects (images 2(a) and 2(b)). The cluster does not contain the front sides (images 2(c) and 2(d)), which are much more different. In fact, there is another cluster, containing the front sides of object 938 only. The F-measure with the individual clusters is just around 0.25, adding the second object information improves this only slightly to 0.285 (due to the bad recall), but when using both clusters and some color features it rises to above 0.9, offering a much better explanation. Note that one might also be tempted to see a shape cluster, while this is impossible due to the result being computed from color histograms. From the perspective of multiview-clustering, this is a very interesting cluster, since there is a nontrivial cluster that is orthogonal to the original classes, consisting only of parts of the original classes each. As such, the automatic analysis also returns the two matching objects along with color annotations for a best match, thus supporting the analysis as intended. Also note that the reported green colors do not resemble the picture much – but the images may indeed have a very similar distance from this reference color. After our initial analysis we added some new

(a) Object 938 (b) Object 939 (c) Object 938 (d) Object 939

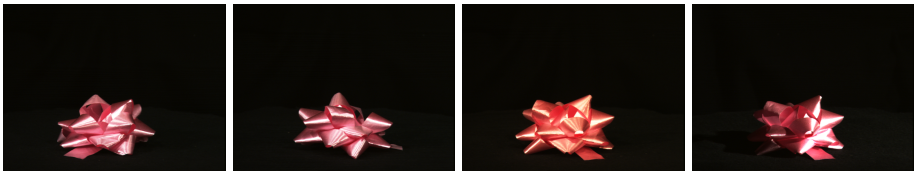
Score	Analysis
0.914	color-408055 color-aaff80 object-938 object-939
0.635	color-808040 color-aaff80 object-938 object-939
0.286	object-938 object-939
0.257	object-939
0.243	object-938
0.877	object-938 object-939 **front-to-back**
0.618	object-938 **front-to-back**
0.590	object-939 **front-to-back**

(e) Analysis result

Fig. 2. Boxes in ALOI image data set

features, including a front-to-back object scoring (ranging from 0 to 1 based on the angle the image was taken from). Including this scoring returns some new explanations. However, they do not score as well as the color-based explanations, making the less interesting color explanation more appropriate. Nevertheless, we already discovered structure in the data that we had not formalized before.

Another cluster (Figure 3) contains 81 images of object 49 and 2 others (so it is almost pure in a traditional sense), but only scores 0.848 on the object itself. Combined with a single color feature, this improves to 0.969. Images 3(a) and 3(b) were both included in the cluster (as were all other rotations and basic color situations). Given the strong uniformity of the object's color representation under rotation, OPTICS cuts off the color variations of the cluster such as image 3(c) (having a light color of $2172K$ as opposed to $3075K$ for the regular images) and angular lighting situations such as image 3(d) (with light coming from the bottom right instead of the center). While the cluster matches the object very well, the actual subset included can be better explained when also using color scorings. Some other objects (e.g. the sea shell 228) were clustered the same way. Furthermore, OPTICS also found a subcluster within this cluster containing just 33 images. The F-measure for the object class on this sub-cluster was just a meagre 0.468. Combining it with a feature that contains only the basic lighting situations, the score rises to 0.606. However, the direct color based explanations match better than the ground truth lighting information, so the clustering algorithm does not appear to have recognized the actual light effects here. In both examples shown so far, the clustering algorithm far from failed: it discovered that there is a *subset* of a class that is more similar to each other than the others.



(a) Regular light (b) Regular light (c) Temp. $2172K$ (d) Light angle $l1c2$

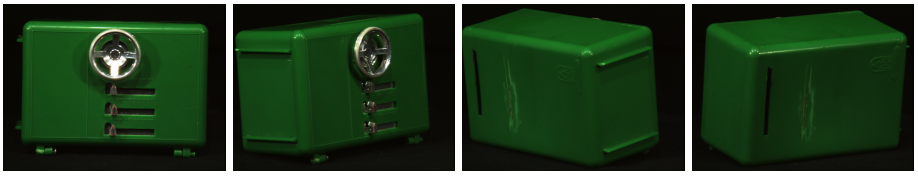
Cluster	Score	Analysis
Parent	0.976	color-2a0080 color-80ff80 object-49
Parent	0.969	color-2a0080 object-49
Parent	0.963	color-aaff00 object-49
Parent	0.848	object-49
Parent	0.936	color-ffd580 object-49 **light-basic**
Parent	0.918	object-49 **light-basic**
Child	0.909	color-008080 color-800080 color-ffd580 object-49
Child	0.899	color-2a0080 color-ffd580 object-49
Child	0.829	color-2a0080 object-49
Child	0.468	object-49
Child	0.925	color-2a0080 color-800080 color-aaff00 object-49 **light-basic**
Child	0.923	color-800080 color-80ff80 object-49 **light-basic**
Child	0.606	object-49 **light-basic**

(e) Analysis of clusters

Fig. 3. Decorative loop in ALOI image data set (Object 49)

In object 492 another interesting hierarchy was discovered (see Figure 4). The outer cluster contains 99 images of the object, while the inner cluster contained just 29. The outer cluster obviously is fairly complete, it just misses some lighting conditions. The inner cluster contains only front views of the object (images 4(a) and 4(b)), but not of the back side (images 4(c) and 4(d)). This is not very surprising, given the silver handle present on the front side of the object, but absent from the back. Note that the inner cluster is explained by colors much better than by the object class, despite being pure, while the outer cluster also scored very well when compared with the object itself. For this cluster again we added the front-to-back object scoring. For the main cluster, this does not improve the result at all (as expected). For the inner cluster, the result however almost doubles, allowing the claim that the algorithm had successfully discovered front views of the object. The result slightly improves with additional color scorings, which is not surprising given that the algorithm had used color information.

Then there is a cluster that caught our attention by having 155 images, making it clearly larger than the expected class size. It contained 74 and 75 images of the objects 981 and 982, respectively, two very similar metal pots (see Figure 5), along with 5 other objects (likely an artifact of the OPTICS ξ "steep up area" definition). The automated analysis suggests that the cluster is based

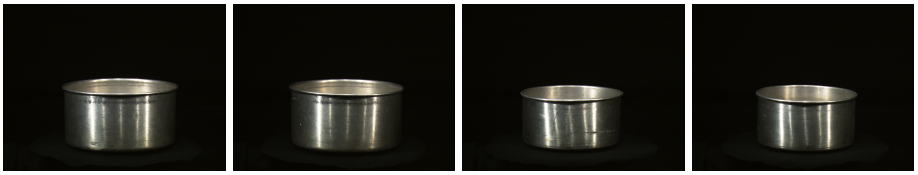


(a) Front of 492 (b) Front of 492 (c) Back of 492 (d) Back of 492

Cluster	Score	Analysis
Parent	0.971	color-00ff00 color-408040 object-492
Parent	0.961	color-00802b color-00ff00 object-492
Parent	0.938	object-492
Parent	0.938	object-492 **front-to-back**
Child	0.812	color-80002b
Child	0.774	color-00802b
Child	0.414	object-492
Child	0.852	color-408040 color-80002b object-492 **front-to-back**
Child	0.846	color-408040 **front-to-back**
Child	0.821	object-492 **front-to-back**

(e) Analysis result

Fig. 4. Green savings box in ALOI image data set



(a) Object 981 (b) Object 981 (c) Object 982 (d) Object 982

Score	Analysis
0.955	color-00802b color-2a8000 object-981 object-982
0.940	color-0000ff object-981 object-982
0.925	color-ffff80 object-981 object-982
0.790	object-981 object-982
0.564	object-982
0.556	object-981
0.974	color-00802b object-981 object-982 **light-basic**
0.939	object-981 object-982 **light-basic**

(e) Analysis result

Fig. 5. Metal pots in ALOI image data set

on the two objects along with color restrictions. However, when adding the basic lighting scoring again, the result is explained better. In retrospection, this is not surprising, given that the metallic object does reflect the light to some extend,

and the object color is thus expected to vary much with the light in contrast to for example the green objects before. Again there is a child cluster and a super cluster which adds 36 images of another metallic object.

4 Conclusion

Building upon some points taken concerning the evaluation of multiple clusterings in last year's MultiClust workshop [3], here we developed some first steps in implementing the vision. We provide a system for evaluation of clusterings, based on prior knowledge as well as on readily extensible knowledge. Currently, the system comprises the ALOI data set. We discussed exemplary clustering results for these data in a case study. The system allows to judge whether some cluster is rather trivial (given it is related to a known concept at all), whether it is a combination of such concepts, or whether it might comprise an interesting, non-trivial, new concept.

During the case study performed on the ALOI data set, we were able to discover nontrivial structure in the data set that we had not been aware of before, but that we were able to formalize and add back into the system to improve the analysis results.

Along with the reference files computed for the ALOI data set, including the advanced structure we found during our analysis, the analysis toolkit is available on the ELKI web page: `http://elki.dbs.ifi.lmu.de/`.

We encourage researchers to use and extend this toolkit for evaluating their findings and for contributing additional structure information. We also would welcome the incorporation of other data sets.

References

1. Kriegel, H.P., Kröger, P., Sander, J., Zimek, A.: Density-based clustering. WIREs DMKD **1**(3) (2011) 231–240
2. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of Eugenics **7** (1936) 179–188
3. Färber, I., Günnemann, S., Kriegel, H.P., Kröger, P., Müller, E., Schubert, E., Seidl, T., Zimek, A.: On using class-labels in evaluation of clusterings. (2010)
4. Zeeberg, B.R., Feng, W., Wang, G., Wang, M.D., Fojo, A.T., Sunshine, M., Narasimhan, S., Kane, D.W., Reinhold, W.C., Lababidi, S., Bussey, K.J., Riss, J., Barrett, J.C., Weinstein, J.N.: GoMiner: a resource for biological interpretation of genomic and proteomic data. Genome Biology **4**(4:R28) (2003)
5. Al-Shahrour, F., Diaz-Uriarte, R., Dopazo, J.: FatiGO: a web tool for finding significant associations of Gene Ontology terms with groups of genes. Bioinformatics **20**(4) (2004) 578–580
6. Datta, S., Datta, S.: Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. BMC Bioinformatics **7**(397) (2006)
7. Gat-Viks, I., Sharan, R., Shamir, R.: Scoring clustering solutions by their biological relevance. Bioinformatics **19**(18) (2003) 2381–2389

8. Gibbons, F.D., Roth, F.P.: Judging the quality of gene expression-based clustering methods using gene annotation. Genome Res. **12** (2002) 1574–1581
9. Lee, S.G., Hur, J.U., Kim, Y.S.: A graph-theoretic modeling on GO space for biological interpretation of gene clusters. Bioinformatics **20**(3) (2004) 381–388
10. Prelić, A., Bleuler, S., Zimmermann, P., Wille, A., Bühlmann, P., Guissem, W., Hennig, L., Thiele, L., Zitzler, E.: A systematic comparison and evaluation of biclustering methods for gene expression data. Bioinformatics **22**(9) (2006) 1122–1129
11. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G.: Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat. Genet. **25**(1) (2000) 25–29
12. Bickel, S., Scheffer, T.: Multi-view clustering. In: Proc. ICDM. (2004)
13. Cui, Y., Fern, X.Z., Dy, J.G.: Non-redundant multi-view clustering via orthogonalization. In: Proc. ICDM. (2007)
14. Jain, P., Meka, R., Dhillon, I.S.: Simultaneous unsupervised learning of disparate clusterings. Stat. Anal. Data Min. **1**(3) (2008) 195–210
15. Günnemann, S., Müller, E., Färber, I., Seidl, T.: Detection of orthogonal concepts in subspaces of high dimensional data. In: Proc. CIKM. (2009)
16. Qi, Z.J., Davidson, I.: A principled and flexible framework for finding alternative clusterings. In: Proc. KDD. (2009)
17. Dang, X.H., Bailey, J.: Generation of alternative clusterings using the CAMI approach. In: Proc. SDM. (2010)
18. Kriegel, H.P., Zimek, A.: Subspace clustering, ensemble clustering, alternative clustering, multiview clustering: What can we learn from each other? (2010)
19. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM TKDD **3**(1) (2009) 1–58
20. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: Knowledge discovery and data mining: Towards a unifying framework. In: Proc. KDD. (1996)
21. Pfitzner, D., Leibbrandt, R., Powers, D.: Characterization and evaluation of similarity measures for pairs of clusterings. KAIS **19**(3) (2009) 361–394
22. Geusebroek, J.M., Burghouts, G.J., Smeulders, A.: The Amsterdam Library of Object Images. Int. J. Computer Vision **61**(1) (2005) 103–112
23. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proc. SIGMOD. (1999)
24. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-Tree: An efficient and robust access method for points and rectangles. In: Proc. SIGMOD. (1990)
25. Swain, M., Ballard, D.: Color indexing. Int. J. Computer Vision **7**(1) (1991) 11–32
26. Achtert, E., Hettab, A., Kriegel, H.P., Schubert, E., Zimek, A.: Spatial outlier detection: Data, algorithms, visualizations. In: Proc. SSTD. (2011)

Browsing Robust Clustering-Alternatives

Martin Hahmann, Dirk Habich, and Wolfgang Lehner

TU Dresden; Database Technology Group; Dresden, Germany
{martin.hahmann, dirk.habich, wolfgang.lehner}@tu-dresden.de

Abstract. In the last years, new clustering approaches utilizing the notion of multiple clusterings have gained attention. Two general directions — each with its individual benefits — are identifiable: (i) extraction of multiple alternative clustering solutions from one dataset and (ii) combination of multiple clusterings of a dataset into one robust consensus-solution. In this paper, we propose a novel hybrid approach to generate and browse robust, alternative clustering results. Our hybrid approach is based on *frequent-groupings* as specialization of frequent-itemset mining. In this way, the different benefits of the existing directions are combined, offering new opportunities for knowledge extraction.

1 Introduction

Depending on which notion you follow, the information age has been around for 20 to 40 years and brought with it a massive trend for digitalization and data collection. Just like coal and steel in the prior age of industrialization, information and data have become a crucial resource for today's work. In contrast to the diminishing natural raw materials, the deposits of data are constantly growing. While data itself is already valuable, it is only capitalized to the full extent if knowledge can be obtained from it. For this task, analysis techniques like clustering have been developed [10]. The goal of clustering is to group a set of data objects into different clusters, so that members of one cluster are similar to each other, while different clusters are dissimilar.

A multitude of clustering algorithms has been proposed over the years [10], whereas these traditional algorithms have several limitations. On the one hand, they are not generally applicable or robust meaning that certain algorithms and parametrizations only suit certain datasets and will yield poor results otherwise. On the other hand, traditional techniques generate only a single clustering, but as todays datasets become more complex and high-dimensional, in general there are more clustering results possible for a dataset. Besides these data centric problems, usability and applicability have become important issues as clustering evolves from a niche application in research to a widespread analysis technique employed in more and more areas. With this trend, new users come into contact with clustering, who are often experts of their respective application domain but have no experience in the area of clustering. This calls for clustering approaches that can be versatilely applied and lack the complicated algorithm selection and configuration of traditional approaches.

In recent years, a number of approaches have been proposed to tackle some of these issues. The area of *alternative clustering* [2, 3] provides multiple clustering solutions for a dataset. With this, several views on complex data can be offered, while the availability of multiple clusterings in the first place, usually frees the user from adjusting a single clustering that proves unsatisfactory. An opposing approach is *ensemble clustering* [11, 4] in which a set of multiple clusterings is integrated to form a single consensus clustering result. This input set is also called clustering-ensemble and contains results that are generated using different algorithms and parameters. The consensus result is often more robust than clusterings generated by a single algorithm and set of parameters, which means that this technique is more versatile in terms of the application scenario. Additionally, algorithm selection and configuration is eased as a range of methods and parameters is utilized. On the downside, ensemble clustering provides just one solution and therefore resembles traditional clustering at that point. To create an alternative clustering solution, the user has to switch and/or re-parametrize the employed clustering algorithms. This is a very challenging task because a set of algorithms must be selected and configured.

To summarize, *alternative clustering* and *ensemble clustering* both have benefits compared to traditional clustering approaches. However, from the user's point of view there is a decision to make. The user must decide if multiple alternative solutions are chosen over one robust solution or vice versa, as it is not possible to have both. In this paper we address this issue by proposing an idea for combining the approaches of alternative and ensemble clustering, that makes the creation of robust alternatives possible. We start with a short description of alternative and ensemble clustering in Section 2. Then, we describe *frequent-groupings* as the core concept our our novel hybrid approach in Section 3. Our *frequent-grouping* technique is based in the idea of frequent-itemset mining [1] and allows the identification of robust clusters, occurring throughout the clustering ensemble. Subsequently, we present how these frequent-groupings can be combined in order to create multiple robust alternatives in Section 4. We outline an algorithm-driven as well as a user-driven approach that enables the creation of alternatives by browsing and switching frequent-groupings. We conclude our paper with a discussion of open issues in Section 5 and a short summary in Section 6.

2 Related Work

The problems of traditional clustering, that we sketched in the introduction often lead to multiple iterations in which different parameters or clustering algorithms are tried out until a satisfactory clustering result is obtained. This trial-and-error practice implicitly generates multiple clustering solutions for the analyzed dataset. Over the last years, new approaches to clustering emerged, that explicitly utilize multiple clustering solutions for the knowledge discovery process. In this section, we briefly review two of these approaches, namely: *alternative clustering*, which provides the user with multiple clustering solutions for a dataset

and *ensemble clustering*, which integrates multiple clusterings of a dataset into a single robust solution.

2.1 Alternative Clustering

The main goal of this clustering approach is to provide alternative clustering solutions to the user. To create alternative solutions, at first an initial clustering result is made, using a traditional clustering algorithm. Based on the information contained in this initial clustering, alternative solutions are generated so that these alternatives are *dissimilar* to the initial solution. As an example for alternative clustering, we describe the *COALA* [2] algorithm. Given a Clustering C with k clusters this method generates a dissimilar alternative S also having k clusters. To express dissimilarity the authors use instance-based 'cannot-link' constraints, that are derived from the initial clustering and are employed in the construction of the alternative. Such a constraint can be expressed as a pair of objects (x_i, x_j) with $i \neq j$. A clustering satisfies this constraint if x_i and x_j are not located in the same cluster. In order to reach the maximum degree of dissimilarity from C, the alternative S should place as much objects as possible in different clusters, that were in the same cluster in C. Although this approach seems plausible, strict adherence to it can lead to meaningless solutions, as a clustering that maximizes dissimilarity most likely does not comply with the general requirements of clustering, namely similar objects belong to the same cluster while clusters are dissimilar. This means besides being dissimilar, an alternative clustering must also satisfy a certain quality, that is, in the case of COALA, expressed by the similarity of a pair of objects. The two goals of dissimilarity and quality can be inversely related, for which case COALA offers a parameter ω to control the trade-off between both. COALA works in an iterative way: the initial clusters contain one object each and are successively merged until k is reached. In each iteration two merge candidates are identified: one cluster pair that pursues the quality goal by having the smallest distance (d_{qual}) of all pairs and one cluster pair pursuing dissimilarity, that has the smallest distance (d_{diss}) of all pairs that fulfill the cannot-link constraints. The decision between both candidates is based on an inequation: if $d_{qual} \leq \omega \cdot d_{diss}$ the quality pair is merged, else the dissimilarity pair. Another alternative clustering technique is CAMI[3] which is also based on the goals of quality and dissimilarity but models them in a different way as it represents clusters using gaussian mixtures.

2.2 Ensemble Clustering

In contrast to alternative clustering, ensemble clustering does not present multiple clusterings to the user, but uses them to generate a single clustering result. This set of multiple clusterings is also called *clustering-ensemble* and contains clustering results generated by executing multiple algorithms with multiple sets of parameters. As is known, certain algorithms or parametrizations do not suit certain datasets, thus producing poor results. By utilizing a wide range of different clustering algorithms and parametrizations, this problem can be tackled.

Thus, the final clustering solution—called *consensus* clustering—generated from such a clustering ensemble is more robust and often has an increased quality [11]. In order to create such a clustering a consensus function is needed, that integrates the cluster assignments of all ensemble members into a new clustering. The use of the term consensus already shows that the goal of this integration is to identify clusters/structures that are detected by the majority of ensemble members and preserve them in the final solution. In other words, similarities throughout the clustering-ensemble are identified and used to create the consensus clustering. Two main classes of ensemble clustering techniques can be distinguished: *pairwise-similarity* approaches and approaches based on *cluster-labels*.

Pairwise-Similarities: Algorithms working on the basis of pairwise similarities model the cluster assignment information by evaluating the assignments of each object pair over the whole ensemble [4, 11]. There are two cases of pairwise similarity: (i) a pair objects is part of the *same* cluster or (ii) a pair of objects is part of *different* clusters. For each local clustering of the ensemble these similarities can be represented in the form of a so called coassociation matrix, in which a cell contains a 1 if the respective pair of objects is located in the same cluster or a 0 if an object pair is assigned to different clusters. By adding up all the local matrices and normalizing each cell using the ensemble size, a global coassociation matrix is build that contains the relative frequency in which each pair of objects is located in the same cluster throughout the whole ensemble. Based on this matrix, different consensus functions can be employed to extract the final solution. As an example. we describe a very simple function based on [4], which generates a consensus clustering on the following basis: if a pair of objects is located in the same cluster in the majority of the ensemble, i.e. at least 50% of the clusterings, it should also be part of the same cluster in the consensus solution. Vice versa this also holds for object pairs mostly located in different clusters. Therefore, the consensus clustering shows minimal dissimilarities to the ensemble in terms of pairwise similarities. The consensus function is realized by removing all cells from the global coassociation matrix that contain a value smaller than 0.5 and use the remaining cells to generate the clustering.

Cluster-Labels: In contrast to pairwise-similarity based approaches, other techniques for ensemble clustering directly use the cluster assignments from the ensemble by working with the provided cluster labels only. As no coassociation matrices are used, they are often less time-consuming. Various algorithms exist in this class e.g. *Ensemble-Merging* [9] which assumes a clustering ensemble having a constant number of clusters k, that are each represented by a centroid. By grouping similar centroids of the ensemble, k global centroids are generated which are then used to build the consensus clustering.

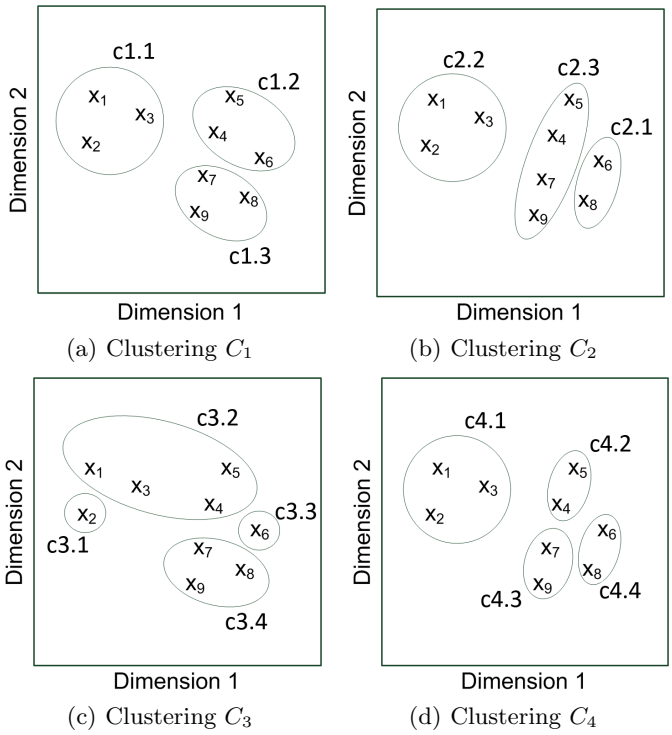
3 Finding Frequent Groupings

In the previous section, we described that ensemble clustering creates a single consensus clustering out of a set of different multiple clusterings. We outlined the two major approaches to this task, namely cluster-labels and pairwise-

similarities. While the techniques of both approaches differ in their mode of operation, they share the common goal to incorporate those parts of the dataset into the final solution, whose cluster assignments agree with the majority of the clustering-ensemble. In other words, clusters of the consensus solution are sets of objects, that were frequently assigned to the same cluster throughout the clustering-ensemble. For the purpose of illustration, we use the small clustering-ensemble depicted in Figure 1 as a running example. Our example contains a dataset $\mathcal{D} = \{x_1, x_2, \ldots, x_9\}$ of nine objects in a $2D$ feature space. For $\mathcal{D}$ exists a clustering-ensemble $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$ that contains four clusterings, each with a different number of clusters and different cluster composition. To model and evaluate the similarities of the cluster assignments of an object in the ensemble, label-based approaches match cluster ids or representations, while approaches based on pairwise-similarities count the co-occurence of object pairs. To give an example for a consensus clustering, we apply the pairwise approach mentioned in Section 2 to $\mathcal{C}$ and obtain a clustering with the clusters $c_1 = \{x_1, x_2, x_3\}$, $c_2 = \{x_4, x_5\}$, and $c_3 = \{x_6, x_7, x_8, x_9\}$. In addition to these two principles, we propose a novel third way to identify the frequent assignment of objects to the same cluster, which is based on the concept of frequent itemsets [1].

Assuming a set of n items $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ and a set of transactions $\mathcal{T} = \{t_1, \ldots, t_m\}$ of which each transaction has a unique id an contains a subset of $\mathcal{I}$, a set of items $\mathcal{X}$ is called frequent if its *support* exceeds a given threshold. The support of an itemset $\mathcal{X}$ is defined by the fraction of transactions of $\mathcal{T}$ that contain $\mathcal{X}$. At this point, the analogy to ensemble clustering should be obvious: while frequent itemsets aim to identify items that co-occur in many transitions while ensemble clustering searches for objects occuring together in the majority of clusters. In the following, we map the concepts of $\mathcal{I}$, $\mathcal{T}$ and *support* to the domain of ensemble clustering in order to describe a method that allows the identification of *frequent-groupings*.

While it is obvious that the items of $\mathcal{I}$ correspond to the objects of the dataset for a clustering, the matching of the transaction concept is more intricate. As $\mathcal{T}$ is a set of transactions, that each contain elements of $\mathcal{I}$. At first sight, it could be mapped to the clustering-ensemble, because the ensemble also consists of multiple clusterings, containing the elements of the dataset. However this mapping should not be used, because it effectively prevents the identification of frequent sets of objects, as in general, all objects of a dataset are assigned to a cluster. Assuming the transaction-clustering analogy, this means that each transaction contains all items of $\mathcal{I}$, which makes it impossible to identify interesting, frequent object groupings. Therefore we model $\mathcal{T}$ as the set of clusters from all members of the clustering-ensemble, as each of them normally only contains a subset of the data. Using the mappings made so far, the *support* of a set of data objects $\mathcal{X}$ shows the fraction of the clustering-ensemble, in which $\mathcal{X}$ is part of the same cluster. If $support(\mathcal{X})$ exceeds a certain threshold, we call $\mathcal{X}$ a *frequent-grouping*. A high support of $\mathcal{X}$ also shows that this set of objects is robust, because it was



(a) Clustering C_1 (b) Clustering C_2

(c) Clustering C_3 (d) Clustering C_4

Fig. 1. The clustering-ensemble of the running example.

identified as part of a cluster in many clusterings, regardless of the employed algorithm and/or parameters.

With regard to our running example, this means that $\mathcal{D}$ acts as itemset, while the clustering-ensemble $\mathcal{C}$ provides a set of 14 clusters/transactions. As transactions are required to have unique identifiers, we label them in a special way e.g. $c1.2$ marks cluster 2 of clustering C_1. To simplify the calculation of support, we make the following assumption: each $x_i \in \mathcal{D}$ is assigned to exactly one cluster in each $C_i \in \mathcal{C}$. With this, an object can only occur in one cluster resp. transaction per clustering. To illustrate what this means, we regard the group of objects (x_1, x_2, x_3) from our running example. These objects occur in the three clusters $c1.1, c2.2$, and $c4.1$ thus this itemset has a support of $3/4$ resp. 0.75 as it occurs in three clusterings C_1, C_2, and C_4.

In order to generate the frequent-groupings from our running example, we first must specify a threshold for the support to decide whether a group of objects occurs frequently or not. Following the assumptions of [4] we regard a set of objects as frequent, if it occurs at least in 50% of the clustering-ensemble, which means two clusterings in our example. The obtained frequent-groupings are depicted in Figure 2 in the form of a graph structure. Each node represents a

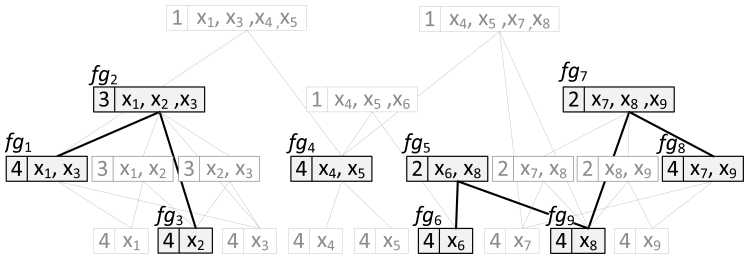


Fig. 2. Structure of the frequent-groupings generated from the running example.

frequent-grouping and contains its associated objects as well as its support, while edges indicate subset/superset relations between nodes. To build this structure we initially create and insert a node for each cluster found in the ensemble. If a node already exists in the graph structure it is not inserted again but the support of the existing node is increased by one e.g. the objects $\{x_1, x_2, x_3\}$ are contained in the clusters $c1.1, c2.2$, and $c4.1$, thus only one node with a support of three is generated. From the initial graph all nodes that are not frequent, i.e. with a support less than two are filtered—e.g. $c3.2$—and are displayed in a faded grey. For each remaining frequent-grouping a new set of nodes is created, containing all of its possible subsets. At last all frequent-grouping nodes that are not *closed* are filtered i.e. each node, having a direct superset with the same support is removed from the graph. Eventually this procedure leads to the nine frequent-groupings $fg_1, fg_2, \ldots, fg_9$ displayed in Figure 2. Please note that the described procedure only illustrates the formation of the depicted graph structure. There already exist sophisticated methods for mining closed frequent itemsets, that can be applied to generate frequent-groupings more efficiently [12].

By interpreting the obtained frequent-groupings as clusters, we can generate a consensus clustering by combining them in a way, that all objects of $\mathcal{D}$ are located in a cluster. As the frequent-groupings overlap, multiple *alternative* combinations can be produced. For our running example, six alternative consensus clusterings can be created, which are shown in Table 1. We will discuss the actual construction of these alternatives in the following section. In contrast to the alternative clusterings generated by existing approaches, our solutions feature a certain degree of robustness, as for each cluster a consensus exists throughout the ensemble, which is defined via the support threshold. Thus our frequent-grouping approach represents a hybrid between alternative and ensemble clustering, that combines the benefits of both domains, namely alternative solutions and robustness. In addition our approach has some advantages over the cluster-label and pairwise-similarity based techniques. Most label based approaches require that the number of clusters in the consensus solution is specified in advance. This is not necessary with pairwise-similarities or frequent-groupings as the number of consensus clusters results from the co-occurrence of objects in the ensemble. Although this characteristic is a similarity between both techniques there is a difference. As pairwise-similarity methods work with the smallest possible group-

A_1	$\{x_1, x_2, x_3\}\,\{x_4, x_5\}\,\{x_6\}\,\{x_7, x_8, x_9\}$
A_2	$\{x_1, x_2, x_3\}\,\{x_4, x_5\}\,\{x_6, x_8\}\,\{x_7, x_9\}$
A_3	$\{x_1, x_2, x_3\}\,\{x_4, x_5\}\,\{x_6\}\,\{x_8\}\,\{x_7, x_9\}$
A_4	$\{x_1, x_3\}\,\{x_2\}\,\{x_4, x_5\}\,\{x_6\}\,\{x_7, x_8, x_9\}$
A_5	$\{x_1, x_3\}\,\{x_2\}\,\{x_4, x_5\}\,\{x_6, x_8\}\,\{x_7, x_9\}$
A_6	$\{x_1, x_3\}\,\{x_2\}\,\{x_4, x_5\}\,\{x_6\}\,\{x_8\}\,\{x_7, x_9\}$

Table 1. Alternative consensus clusterings of the running example.

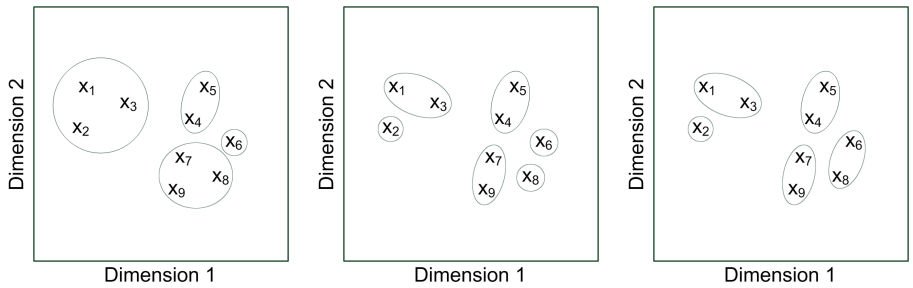
ings i.e. pairs, they are prone to transitive effects. Assume two object pairs (p, q) and (q, r), each one occuring in the same cluster in 50% of the ensemble. Such a setting can lead to (p, q, r) being placed in the same cluster of the consensus solution, even if (p, r) never occurs in the same cluster in the whole ensemble. This cannot happen with frequent-groupings as it evaluates co-occurrence in a different way.

4 Browsing Alternatives

Aside from being able to identify frequent-groupings, we need a combination procedure to generate alternative solutions. The challenging part here is the high combinatorial diversity resulting from the concept of frequent-itemsets and our idea of combination. To illustrate this issue we again look at our running example $\mathcal{D}$, which contains nine objects. All possible frequent-groupings that can occur in a dataset D are contained in the *power set* of D. As we do not need the empty set, there are $2^{|D|} - 1$ possible groupings—i.e. 511 for our example—to begin with. This extremely high number indicates the general scale, we are dealing with but has no direct impact on our approach, as we only consider the groupings that were found in the ensemble and their respective subsets. Besides the reduction by clustering, further candidates for frequent-groupings are removed via the support threshold and the condition that frequent-groupings must be closed. Notably this last filter criterion is important as it keeps the number of small and singleton frequent-groupings low. These small groupings inflate the number of possible alternative consensus clusterings by allowing additional combinations that differ only in the assignment of one or two objects. For our running example this is neglectable but for larger data, this issue must be considered i.e. too small frequent-groupings must be removed. In section 3, we have assumed that each object is assigned to one cluster in each clustering of the ensemble, which means that potentially $|D|$ singleton groupings with a support of 100% exist, each containing one element of D. If these are not considered, the number of alternative solutions rises and even the trivial solution of a clustering that assigns each object to its own cluster is possible.

Having discussed the necessity of frequent-grouping filtering we are now faced with the question of obtaining different alternative clustering solutions. To get all possible alternatives it would be necessary to generate all possible combinations of frequent-groupings—the power set of the set of frequent-groupings—and select

all combinations that contain D and only consist of disjoint frequent-groupings. Obviously this approach is very expensive, thus we propose the use of a greedy approach for the construction of alternatives. By varying the optimization criterion a set of alternatives can be generated. Using the frequent-groupings shown in Figure 2, we extracted the three alternatives depicted in Figure 3 by using three different selection criterions. The alternative in Figure 3(a) was generated with the goal of maximizing cluster size, therefore it contains the frequent-groupings fg_2, fg_4, fg_6, fg_7 and matches alternative A_1 from Table 1. Furthermore, alternative A_6 of Figure 3(b) maximizes the support, while A_5 of Figure 3(c) aims to maximize the number of equal-sized clusters. Naturally it is possible to employ other optimization goals than frequent-grouping size and support. The identification of such goals and more complex greedy heuristics is a topic for future research.



(a) A_1: maximize cluster size (b) A_6: maximize support (c) A_5: equal cluster size

Fig. 3. Alternative consensus clusterings extracted with different greedy approaches.

Using greedy approaches to extract alternatives means that the user needs to specify the number of alternatives, a number of greedy heuristics, and maybe additional parameters. Regarding different levels of user experience and the characteristics of different application domains, this can be a challenging task. Therefore, we propose a second way to generate robust alternatives that allows users to actually browse through alternatives using high-level feedback. This approach relies on our previous work described in [5], where we propose a feedback-driven process for ensemble clustering that allows users to iteratively refine a consensus clustering using a visual-interactive interface [6, 7] and a special paiwise-similarity based ensemble clustering approach [8]. Our proposed process provides an initial clustering solution, which the user interpretes in terms of intra-cluster composition and inter-cluster relations via the proposed visual-interactive interface. Depending on the users evaluation, the initial result can be adjusted using a set of four feedback operations: *merge, split, refine, and restructure*. In the following we transfer the general idea of this process and its feedback options to the setting of our frequent-groupings approach. Therefore we assume

that the user is provided with an initial alternative, created by an arbitrary greedy approach. Furthermore we assume that access to some sort of clustering visualization is available. In this setting the user can now use the four feedback operations to browse through the structure given by our frequent-groupings and their subset/superset relations. In doing so, the user can create different robust alternative clusterings. Subsequently, we use our running example to illustrate this feedback-driven browsing and describe the implementation of our feedback operations in this context. An overview of the implementation is depicted in Figure 4. As intial consensus clustering, we assume alternative A_5 shown in Figure 3(c). Should the user not be satisfied with the clusters fg_1, fg_3 it is possible to combine them into one cluster by applying the *merge* feedback operation to both. To realize the merge, the superset relations resp. the ascending edges of fg_1 and fg_3 are checked inside the graph structure. If these relations meet in a frequent-grouping that is a superset of the originating clusters and equals their union, both original clusters are replaced by the new found superset. In our running example this requirement is fulfilled, thus fg_1 and fg_3 are replaced by fg_2 which effectively transforms A_5 into A_2. By issuing this simple operation the user has generated a new robust alternative. If there exists no suitable superset, then *merge* is not applicable. Based on this, the *split* operation is implemented by traversing the subsets resp. the descending edges of a frequent-grouping towards the nearest set of frequent-groupings, that are subsets of the original grouping and exactly contain all of its members. In our example, the *split* of fg_5 would replace this frequent-grouping with fg_6 and fg_8, thus transforming A_5 into A_6. Again, if there are no suitable subsets—e.g. for fg_4—then a split is not possible. By analyzing the frequent-groupings in advance, availability of split an merge operations can be determined for each frequent-grouping and displayed in the visualization.

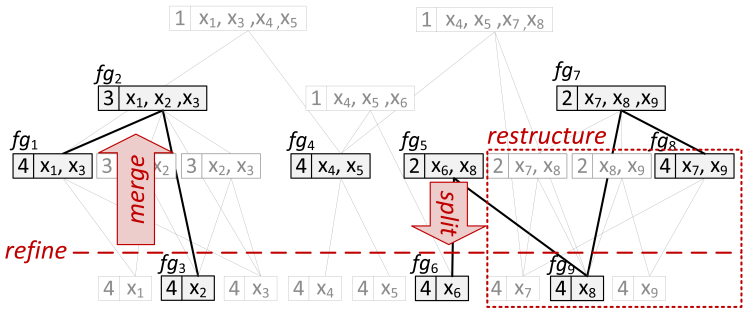


Fig. 4. Implementation of high-level feedback for browsing frequent-groupings.

The remaining feedback operations *refine* and *restructure* are always applicable as they do not navigate through the frequent-groupings but are used to remove or re-create them. With *refine*, frequent-groupings that do not exceed a

certain size are filtered. This allows the removal of clusters considered too small, in the respective application domain and the already mentioned singletons, thus reducing the number of available alternative consensus clusterings. Applying refine in order to eliminate singletons in our running example results in the pruning of fg_3, fg_6, and fg_9, and thus would reduce the available alternatives to A_2. Although this seems like a harsh reduction in our small scale example this method has its merits for larger datasets. Furthermore, use of the refine operation can result in objects of D not being covered by any frequent-grouping. In this case these objects can be considered as *noise* in the consensus clustering, as they cannot be assigned to a robust cluster of significant size. This notion of noise also influences the number of available alternatives which we will demonstrate using the already mentioned application of refine to the running example. If we rule out the existence of noise, the only possible consensus clustering is A_2. On the other hand, the existence of noise makes new alternatives possible, for example A_1 would become $\{x_1, x_2, x_3\}\{x_4, x_5\}\{x_7, x_8, x_9\}$ with x_6 being noise, thus presenting a clustering solution that not only contains robust clusters but also identifies those parts of the dataset for which satisfactory consensus cannot be found. Based on this, it could be possible to draw conclusions regarding dataset structure and pre-processing. Therefore the handling and implications of this kind of noise will be part of our future research.

At last, *restructure* allows the reconfiguration of certain frequent-groupings. Lets assume that a user is especially interested in a specific area of the dataset, but the available frequent-groupings provide no or not enough alternatives resp. split/merge possibilities for this area. In this case *restructure* builds a new clustering-ensemble and new frequent-groupings for the specified subset of D. Application of this operation to fg_7 means for example, that fg_7, fg_8, and fg_9 are replaced by the frequent-groupings, resulting from a new clustering-ensemble generated for $\{x_7, x_8, x_9\}$. The restructure operation should only be used if, based on domain knowledge, other frequent-groupings can be expected in the respective area.

5 Open Issues

Regarding the generation of frequent-groupings, existing algorithms for frequent itemset mining should be adapted to this domain, focusing especially on pruning techniques, measures for interestingness and the influence of different support thresholds. Other areas of interest are the creation of frequent-groupings from fuzzy clustering-ensembles and more generally the evaluation of support without the assumptions, that an object is always assigned to exactly one cluster in each clustering.

For the automatic extraction of robust alternatives, it is necessary to develop additional greedy heuristics. These heuristics should incorporate the notions of quality and dissimilarity that are found in many existing alternative-clustering approaches into the extraction process in an advatageous way. As frequent-groupings and their superset/subset relations can be interpreted as nodes and

edges of a graph, methods for graph-partitioning also promise to be a viable option for the creation of alternative clustering solutions. Additionally, the notion of noise introduced in the previous section must be explored further. On the one hand, methods for filtering too small frequnet-groupings must be developed in order to keep the number of robust alternative clusterings manageable. On the other hand, the meaning of this noise i.e. of objects for which no satisfying consensus can be found, needs to be explored further, as this could allow conclusions regarding the pre-processing of the data or the fit between dataset and employed algorithms/parameters.

Regarding the proposed user-driven browsing of robust alternatives there are open issues like the integration of specific information like support or relations between frequent-groupings into our visual-interactive interface. Furthermore it is necessary to find a way to handle large numbers of alternatives i.e. methods for communicating the availability of many alternatives to the user must be found. In additon the stepping for navigating through alternatives must be chosen adequately. We are positive that our frequent-groupings approach offers many additional opportunities for further research.

6 Summary

In this paper we proposed our idea of combining the concepts of alternative and ensemble-clustering. Based on the well-known technique of frequent itemset mining, we introduced our notion of frequent-clusters as robust/frequently occuring parts of the clustering ensemble. After describing the creation process of frequent-clusters, we proposed an algorithmic and an user-driven approach for the construction of alternative consensus-clusterings from frequent-clusters. The algorithmic approach uses greedy extraction methods and different optimization goals and thus needs to be parametrized by the user. The user-driven browsing on the other hand employs a small set of high-level feedback options, that allows users to navigate and adjust frequent-clusters in order to compose different consensus-clusterings. The clustering results that are obtained with our approach combine benefits from the domains of alternative and ensemble-clustering, namely multiple alternative solutions that are robust.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. pages 487–499, 1994.
2. E. Bae and J. Bailey. Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006)*, pages 53–62, 2006.
3. X. H. Dang and J. Bailey. Generation of alternative clusterings using the cami approach. In *Proceedings of the Tenth SIAM International Conference on Data Mining*, pages 118–129, 2010.
4. A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. In *Proc. of ICDE*, 2005.

5. M. Hahmann, D. Habich, and W. Lehner. Evolving ensemble-clustering to a feedback-driven process. In *Proceedings of the IEEE ICDM Workshop on Visual Analytics and Knowledge Discovery (VAKD)*, 2010.
6. M. Hahmann, D. Habich, and W. Lehner. Visual decision support for ensemble-clustering. In *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM)*, 2010. (to appear).
7. M. Hahmann, D. Habich, and W. Lehner. Touch it, mine it, view it, shape it. In *Proceedings der 14. GI-Fachtagung für Datenbanksysteme in Business, Technology und Web (BTW 2011, February 28 - March 4 2011, Kaiserslautern, Germany)*, 2011.
8. M. Hahmann, P. Volk, F. Rosenthal, D. Habich, and W. Lehner. How to control clustering results? flexible clustering aggregation. In *Advances in Intelligent Data Analysis VIII*, pages 59–70, 2009.
9. P. Hore, L. Hall, and D. Goldgof. A cluster ensemble framework for large data sets. In *IEEE International Conference on Systems, Man, and Cybernetics*, 2006.
10. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3), 1999.
11. A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3, 2002.
12. M. J. Zaki and C. jui Hsiao. Charm: An efficient algorithm for closed itemset mining. pages 457–473, 2002.

Generating a Diverse Set of High-Quality Clusterings*

Jeff M. Phillips, Parasaran Raman, and Suresh Venkatasubramanian

School of Computing, University of Utah
{jeffp,praman,suresh}@cs.utah.edu

Abstract. We provide a new framework for generating multiple good quality partitions (clusterings) of a single data set. Our approach decomposes this problem into two components, generating many high-quality partitions, and then grouping these partitions to obtain k representatives. The decomposition makes the approach extremely modular and allows us to optimize various criteria that control the choice of representative partitions.

1 Introduction

Clustering is a critical tool used to understand the structure of a data set. There are many ways in which one might partition a data set into representative clusters, and this is demonstrated by the huge variety of different algorithms for clustering [9], [35], [8], [27], [45], [31], [43], [14], [28].

Each clustering method identifies different kinds of structure in data, reflecting different desires of the end user. Thus, a key exploratory tool is identifying a diverse and meaningful collection of partitions of a data set, in the hope that these distinct partitions will yield different insights about the underlying data.

Problem specification. The input to our problem is a single data set X. The output is a set of k partitions of X. A *partition* of X is a set of subsets $\mathcal{X}_i = \{X_{i,1}, X_{i,2}, \ldots, X_{i,s}\}$ where $X = \bigcup_{j=1}^{s} X_{i,j}$ and for all j, j' $X_{i,j} \cap X_{i,j'} = \emptyset$. Let $\mathcal{P}_X$ be the space of all partitions of X; since X is fixed throughout this paper, we just refer to this space as $\mathcal{P}$.

There are two quantities that control the nature of the partitions generated. The *quality* of a partition, represented by a function $Q : \mathcal{P} \to \mathbb{R}^+$, measures the degree to which a particular partition captures intrinsic structure in data; in general, most clustering algorithms that identify a single clustering attempt to optimize some notion of quality. The *distance* between partitions, represented by the function $d : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$, is a quantity measuring how dissimilar two partitions are. The partitions $\mathcal{X}_i \in \mathcal{P}$ that do a better job of capturing the structure of the data set X will have a larger quality value $Q(\mathcal{X}_i)$. And the partitions $\mathcal{X}_i, \mathcal{X}_{i'} \in \mathcal{P}$ that are more similar to each other will have a smaller distance value $d(\mathcal{X}_i, \mathcal{X}_{i'})$.

* This research was partially supported by NSF award CCF-0953066 and a subaward to the University of Utah under NSF award 0937060 to the Computing Research Association.

A good set of diverse partitions all have large distances from each other and all have high quality scores.

Thus, the goal is this paper is to *generate a set of k partitions that best represent all high-quality partitions as accurately as possible.*

Related Work. There are two main approaches in the literature for computing many high-quality, diverse partitions. However, both approaches focus only on a specific subproblem. *Alternate clustering* focuses on generating one additional partition of high-quality that should be far from a given set (typically of size one) of existing partitions. *k-consensus clustering* assumes an input set of many partitions, and then seeks to return k representative partitions.

Most algorithms for generating alternate partitions [38, 16, 6, 5, 13, 21, 12] operate as follows. Generate a single partition using a clustering algorithm of choice. Next, find another partition that is both far from the first partition and of high quality. Most methods stop here, but a few methods try to discover more alternate partitions; they repeatedly find new, still high-quality, partitions that are far from all existing partitions. This effectively produces a variety of partitions, but the quality of each successive partition degrades quickly.

Although there are a few other methods that try to discover alternate partitions simultaneously [10, 29, 37], they are usually limited to discovering two partitions of the data. Other methods that generate more than just two partitions either randomly weigh the features or project the data onto different subspaces, but use the same clustering technique to get the alternate partitions in each round. Using the same clustering technique tends to generate partitions with clusters of similar shapes and might not be able to exploit all the structure in the data.

The problem of k-consensus, which takes as input a set of $m \gg k$ partitions of a single data set to produce k distinct partitions, has not been studied as extensively. To obtain the input for this approach, either the output of several distinct clustering algorithms, or the output of multiple runs of the same randomized algorithm with different initial seeds are considered [46, 47]. This problem can then be viewed as a clustering problem; that is, finding k clusters of partitions from the set of input partitions. Therefore, there are many possible optimization criteria or algorithms that could be explored for this problem as there are for clustering in general. Most formal optimization problems are intractable to solve exactly, making heuristics the only option. Furthermore, no matter the technique, the solution is only as good as the input set of partitions, independent of the optimization objective. In most k-consensus approaches, the set of input partitions is usually not diverse enough to give a good solution.

In both cases, these subproblems avoid the full objective of constructing a diverse set of partitions that represent the *landscape of all high-quality partitions.* The alternate clustering approach is often too reliant on the initial partition, has had only limited success in generalizing the initial step to generate k partitions. The k-consensus partitioning approach does not verify that its input represents the space of all high-quality partitions, so a representative set of those input partitions is not necessarily a representative set of all high-quality partitions.

Our approach. To generate multiple good partitions, we present a new paradigm which decouples the notion of distance between partitions and the quality of partitions. Prior methods that generate multiple diverse partitions cannot explore the space of partitions entirely since the distance component in their objective functions biases against partitions close to the previously generated ones. These could be interesting partitions that might now be left out. To avoid this, we will first look at the space of all partitions more thoroughly and then pick non-redundant partitions from this set. Let k be the number of diverse partitions that we seek. Our approach works in two steps. In the first step called the *generation step*, we first sample from the space of all partitions proportional to their quality. Stirling numbers of the second kind, $S(n, s)$ is the number of ways of partitioning a set of n elements into s nonempty subsets. Therefore, this is the size of the space that we sample from. We illustrate the sampling in figure 1. This generates a set of size $m \gg k$ to ensure we get a diverse sample that represents the space of all partitions well, since generating only k partitions in this phase may "accidentally" miss some high quality region of $\mathcal{P}$. Next, in the *grouping step*, we cluster this set of m partitions into k sets, resulting in k clusters of partitions. We then return one representative from each of these k clusters as our output alternate partitions.

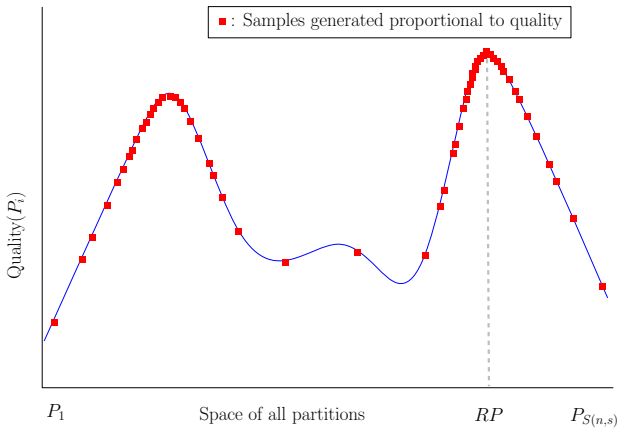


Fig. 1. Sampling partitions proportional to its quality from the space of all partitions with s clusters.

Note that because the *generation step* is decoupled from the *grouping step*, we treat all partitions fairly, independent of how far they are from the existing partitions. This allows us to explore the true density of high quality partitions in $\mathcal{P}$ without interference from the choice of initial partition. Thus, if there is a dense set of close interesting partitions our approach will recognize that. Also because the *grouping step* is run separate from the *generation step*, we can abstract this problem to a generic clustering problem, and we can choose one of many approaches. This allows us to capture different properties of the diversity

of partitions, for instance, either guided just by the spatial distance between partitions, or also by a density-based distance which only takes into account the number of high-quality partitions assigned to a cluster.

From our experimental evaluation, we note that decoupling the generation step from the grouping step helps as we are able to generate a lot of very high quality partitions. In fact, the quality of some of the generated partitions is better than the quality of the partition obtained by a consensus clustering technique called LiftSSD [39]. The relative quality w.r.t. the reference partition of a few generated partitions even reach close to 1. To our best knowledge, such partitions have not been uncovered by other previous meta-clustering techniques. The grouping step also picks out representative partitions far-away from each other. We observe this by computing the closest-pair distance between representatives and comparing it against the distance values of the partitions to their closest representative.

Outline. In Section 2, we discuss a sampling-based approach for generating many partitions proportional to their quality; i.e. the higher the quality of a partition, the more likely it is to be sampled. In Section 3, we describe how to choose k representative partitions from the large collection partitions already generated. We will present the results of our approach in Section 4. We have tested our algorithms on a synthetic dataset, a standard clustering dataset from the UCI repository and a subset of images from the Yale Face database B.

2 Generating Many High Quality Partitions

In this section we describe how to generate many high quality partitions. This requires (1) a measure of quality, and (2) an algorithm that generates a partition with probability proportional to its quality.

2.1 Quality of Partitions

Most work on clustering validity criteria look at a combination of how compact clusters are and how separated two clusters are. Some of the popular measures that follow this theme are *S_Dbw*, *CDbw*, *SD* validity index, maximum likelihood and Dunn index [23, 24, 36, 44, 15, 4, 32, 17]. Ackerman et. al. also discuss similar notions of quality, namely *VR* (variance ratio) and *WPR* (worst pair ratio) in their study of clusterability [1, 2]. We briefly describe a few specific notions of quality below.

k-Means quality. If the elements $x \in X$ belong to a metric space with an underlying distance $\delta : X \times X \to \mathbb{R}$ and each cluster $X_{i,j}$ in a partition $\mathcal{X}_i$ is represented by a single element $\bar{x}_j$, then we can measure the inverse quality of a cluster by $\bar{q}(X_{i,j}) = \sum_{x \in X_{i,j}} \delta(x, \bar{x}_j)^2$. Then the quality of the entire partition is then the inverse of the sum of the inverse qualities of the individual clusters: $\bar{Q}(\mathcal{X}_i) = 1/(\sum_{j=1}^{s} \bar{q}(X_{i,j}))$.

This corresponds to the quality optimized by s-mean clustering [1], and is quite popular, but is susceptible to outliers. If all but one element of X fit neatly in s clusters, but the one remaining point is far away, then this one point dominates the cost of the clustering, even if it is effectively noise. Specifically, the quality score of this measure is dominated by the points which fit least well in the clusters, as opposed to the points which are best representative of the true data. Hence, this quality measure may not paint an accurate picture about the partition.

Kernel distance quality. We introduce a method to compute quality of a partition, based on the kernel distance [30]. Here we start with a similarity function between two elements of X, typically in the form of a (positive definite) kernel: $K : X \times X \to \mathbb{R}^+$. If $x_1, x_2 \in X$ are more similar, then $K(x_1, x_2)$ is smaller than if they are less similar. Then the overall similarity score between two clusters $X_{i,j}, X_{i,j'} \in \mathcal{X}_i$ is defined $\kappa(X_{i,j}, X_{i,j'}) = \sum_{x \in X_{i,j}} \sum_{x' \in X_{i,j'}} K(x, x')$, and a single clusters self-similarity for $X_{i,j} \in \mathcal{X}_i$ is defined $\kappa(X_{i,j}, X_{i,j})$. Finally, the overall quality of a partition is defined $Q_K(\mathcal{X}_i) = \sum_{j=1}^{s} \kappa(X_{i,j}, X_{i,j})$.

If X is a metric space, the highest quality partitions divide X into s Voronoi cells around s points – similar to s-means clustering. However, its score is dominated by the points which are a good fit to a cluster, rather than outlier points which do not fit well in any cluster. This is a consequence of how kernels like the Gaussian kernel taper off with distance, and is the reason we recommend this measure of cluster quality in our experiments.

2.2 Generation of Partitions Proportional to Quality

We now discuss how to generate a sample of partitions proportional to their quality. This procedure will be independent of the measure of quality used, so we will generically let $Q(\mathcal{X}_i)$ denote the quality of a partition. Now the problem becomes to generate a set $Y \subset \mathcal{P}$ of partitions where each $\mathcal{X}_i \in Y$ is drawn randomly proportionally to $Q(\mathcal{X}_i)$.

The standard tool for this problem framework is a Metropolis-Hastings random-walk sampling procedure [34, 25, 26]. Given a domain X to be sampled and an energy function $Q : X \to \mathbb{R}$, we start with a point $x \in X$, and suggest a new point x_1 that is typically "near" x. The point x_1 is accepted unconditionally if $Q(x_1) \geq Q(x)$, and is accepted with probability $Q(x_1)/Q(x)$ if not. Otherwise, we say that x_1 was *rejected* and instead set $x_1 = x$ as the current state. After some sufficiently large number of such steps t, the expected state of x_t is a random draw from $\mathcal{P}$ with probability proportional to Q. To generate many random samples from $\mathcal{P}$ this procedure is repeated many times.

In general, Metropolis-Hastings sampling suffers from high autocorrelation, where consecutive samples are too close to each other. This can happen when

[1] it is commonplace to use k in place of s, but we reserve k for other notions in this paper

far away samples are rejected with high probability. To counteract this problem, often Gibbs sampling is used [41]. Here, each proposed step is decomposed into several orthogonal suggested steps and each is individually accepted or rejected in order. This effectively constructs one longer step with a much higher probability of acceptance since each individual step is accepted or rejected independently. Furthermore, if each step is randomly made proportional to Q, then we can always accept the suggested step, which reduces the rejection rate.

Metropolis-Hastings-Gibbs sampling for partitions. The Metropolis-Hastings procedure for partitions works as follows. Given a partition $\mathcal{X}_i$, we wish to select a random subset $Y \subset X$ and randomly reassign the elements of Y to different clusters. If the size of Y is large, this will have a high probability of rejection, but if Y is small, then the consecutive clusters will be very similar. Thus, we use a Gibbs-sampling approach. At each step we choose a random ordering σ of the elements of X. Now, we start with the current partition $\mathcal{X}_i$ and choose the first element $x_{\sigma(1)} \in X$. We assign $x_{\sigma(1)}$ to each of the s clusters generating s suggested partitions $\mathcal{X}_i^j$ and calculate s quality scores $q_j = Q(\mathcal{X}_i^j)$. Finally, we select index j with probability q_j, and assign $x_{\sigma(1)}$ to cluster j. Rename the new partition as $\mathcal{X}_i$. We repeat this for all points in order. Finally, after all elements have been reassigned, we set $\mathcal{X}_{i+1}$ to be the resulting partition.

Note that auto-correlation effects may still occur since we tend to have partitions with high quality, but this effect will be much reduced. Note that we do not have to run this entire procedure each time we need a new random sample. It is common in practice to run this procedure for some number t_0 (typically $t_0 = 1000$) of *burn-in* steps, and then use the next m steps as m random samples from $\mathcal{P}$. The rationale is that after the burn-in period, the induced Markov chain is expected to have mixed, and so each new step would yield a random sample from the stationary distribution.

3 Grouping the Partitions

Having generated a large collection Z of $m \gg k$ high-quality partitions from $\mathcal{P}$ by random sampling, we now describe a grouping procedure that returns k representative partitions from this collection. We will start by placing a metric structure on $\mathcal{P}$. This allows us to view the problem of grouping as a metric clustering problem. Our approach is independent of any particular choice of metric; obviously, the specific choice of distance metric and clustering algorithm will affect the properties of the output set we generate. There are many different approaches to comparing partitions. While our approach is independent of the particular choice of distance measure used, we review the main classes.

Membership-based distances. The most commonly used class of distances is membership-based. These distances compute statistics about the number of *pairs* of points which are placed in the same or different cluster in both partitions, and return a distance based on these statistics. Common examples include the Rand distance, the variation of information, and the normalized mutual information [33, 40, 42, 7]. While these distances are quite popular, they ignore information

about the spatial distribution of points within clusters, and so are unable to differentiate between partitions that might be significantly different.

Spatially-sensitive distances. In order to rectify this problem, a number of *spatially-aware* measures have been proposed. In general, they work by computing a concise representation of each cluster and then use the earthmover's distance (EMD) [20] to compare these sets of representatives in a spatially-aware manner. These include CDistance [11], d_{ADCO} [6], CC distance [48], and LiftEMD [39]. As discussed in [39], LiftEMD has the benefit of being both efficient as well as a well-founded metric, and is the method used here.

Density-based distances. The partitions we consider are generated via a sampling process that samples more densely in high-quality regions of the space of partitions. In order to take into account dense samples in a small region, we use a *density-sensitive* distance that intuitively spreads out regions of high density. Consider two partitions $\mathcal{X}_i$ and $\mathcal{X}_{i'}$. Let $d : \mathcal{P} \times \mathcal{P} \to \mathbb{R}^+$ be any of the above natural distances on $\mathcal{P}$. Then let $d_Z : \mathcal{P} \times \mathcal{P} \to \mathbb{R}^+$ be a density-based distance defined as $d_Z(\mathcal{X}_i, \mathcal{X}_{i'}) = |\{\mathcal{X}_l \in Z \mid d(\mathcal{X}_i, \mathcal{X}_l) < d(\mathcal{X}_i, \mathcal{X}_{i'})\}|$.

3.1 Clusters of Partitions

Once we have specified a distance measure to compare partitions, we can cluster them. We will use the notation $\phi(\mathcal{X}_i)$ to denote the representative partition $\mathcal{X}$ is assigned to. We would like to pick k representative partitions, and a simple algorithm by Gonzalez [22] provides a 2-approximation to the best clustering that minimizes the *maximum* distance between a point and its assigned center. The algorithm maintains a set of centers $k' < k$ in C. Let $\phi_C(\mathcal{X}_i)$ represent the partition in C closest to $\mathcal{X}_i$ (when apparent we use just $\phi(\mathcal{X}_i)$ in place of $\phi_C(\mathcal{X}_i)$). The algorithm chooses $\mathcal{X}_i \in Z$ with maximum value $d(\mathcal{X}_i, \phi(\mathcal{X}_i))$. It adds this partition $\mathcal{X}_i$ to C and repeats until C contains k partitions. We run the Gonzalez method to compute k representative partitions using LiftEMD between partitions. We also ran the method using the density based distance derived from using LiftEMD. We got very similar results in both cases and we will only report the results from using LiftEMD in section 4. We note that other clustering methods such as k-means and hierarchical agglomerative clustering yield similar results.

4 Experimental Evaluation

In this section, we show the effectiveness of our technique in generating partitions of good divergence and its power to find partitions with very high quality, well beyond usual consensus techniques.

Data. We created a synthetic dataset *2D5C* with 100 points in 2-dimensions, for which the data is drawn from 5 Gaussians to produce 5 visibly separate clusters. We also test our methods on the Iris dataset containing 150 points in

4 dimensions from UCI machine learning repository [18]. We also use a subset of the Yale Face Database B [19] (90 images corresponding to 10 persons and 9 poses in the same illumination). The images are scaled down to 30x40 pixels.

Methodology. For each dataset, we first run k-means to get the first partition with the same number of clusters specified by the reference partition. Using this as a seed, we generate $m = 4000$ partitions after throwing away the first 1000 of them. We then run the Gonzalez k-center method to find 10 representative partitions. We associate each of the 3990 remaining partitions with the closest representative partition. We compute and report the quality of each of these representative partitions. We also measure the LiftEMD distance to each of these partitions from the reference partition. For comparison, we also plot the quality of consensus partitions generated by LiftSSD [39] using inputs from k-means, single-linkage, average-linkage, complete-linkage and Ward's method.

4.1 Performance Evaluation

Evaluating partition diversity. We can evaluate partition diversity by determining how close partitions are to their chosen representatives using LiftEMD. Low LiftEMD values between partitions will indicate redundancy in the generated partitions and high LiftEMD values will indicate good partition diversity. The resulting distribution of distances is presented in Figures 2(a), 2(b), 2(c), in which we also mark the distance values between a representative and its closest other representative with *red* squares. Since we expect that the representative partitions will be far from each other, those distances provide a baseline for distances considered large. For all datasets, a majority of the partitions generated are generally far from the closest representative partition. For instance, in the Iris data set (2(a)), about three-fourths of the partitions generated are far away from the closest representative with LiftEMD values ranging between 1.3 and 1.4.

Evaluating partition quality. Secondly, we would like to inspect the quality of the partitions generated. Since we intend the generation process to sample from the space of all partitions proportional to the quality, we hope for a majority of the partitions to be of high quality. The ratio between the kernel distance quality Q_K of a partition to that of the reference partition gives us a fair idea of the relative quality of that partition, with values closer to 1 indicating partitions of higher quality. The distribution of quality is plotted in Figures 3(a)3(b)3(c). We observe that for all the datasets, we get a normally distributed quality distribution with a mean value between 0.62 and 0.8. In addition, we compare the quality of our generated partitions against the consensus technique LiftSSD. We mark the quality of the representative partitions with *red* squares and that of the consensus partition with a *blue* circle. For instance, chart 3(a) shows that the relative quality w.r.t. the reference partition of three-fourths of the partitions is better than that of the consensus partition. For the Yale Face data, note that we have two reference partitions namely *by pose* and *by person* and we chose the partition *by person* as the reference partition due to its superior quality.

Visual inspection of partitions. We ran multi-dimensional scaling [3] on the all-pairs distances between the 10 representatives for a visual representation of the space of partitions. We compute the variance of the distances of the partitions associated with each representative and draw Gaussians around them to depict the *size* of each cluster of partitions. For example, for the Iris dataset, as we can see from chart 4(a), the clusters of partitions are well-separated and are far from the original reference partition. In figure 5, we show two interesting representative partitions on the Yale face database. We show the mean image from each of the 10 clusters. Figure 5(a) is a representative partition very similar to the partition *by person* and figure 5(b) resembles the partition *by pose*.

5 Conclusion

In this paper we introduced a new framework to generate multiple non-redundant partitions of good quality. Our approach is a two stage process: in the *generation* step, we focus on sampling a large number of partitions from the space of all partitions proportional to the quality and in the *grouping* step, we identify k representative partitions that best summarizes the space of all partitions.

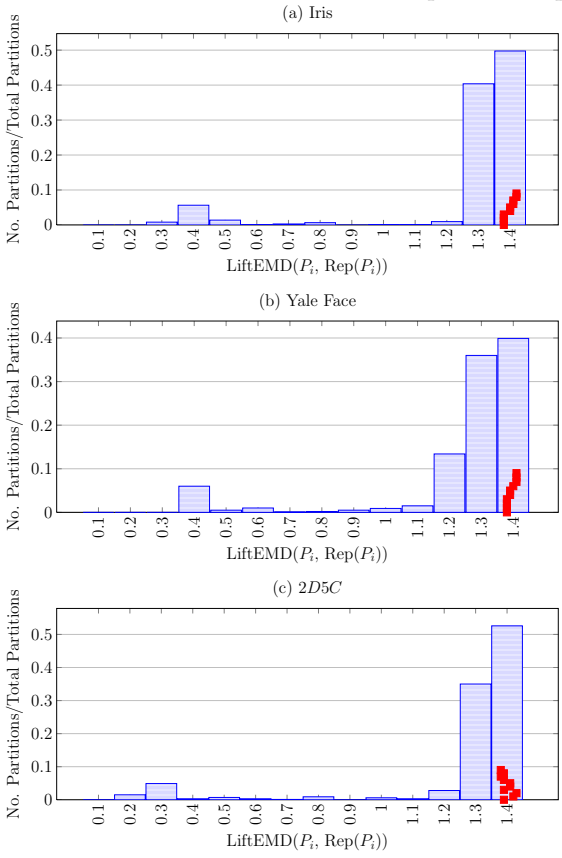


Fig. 2. Distance between partition and its representative.

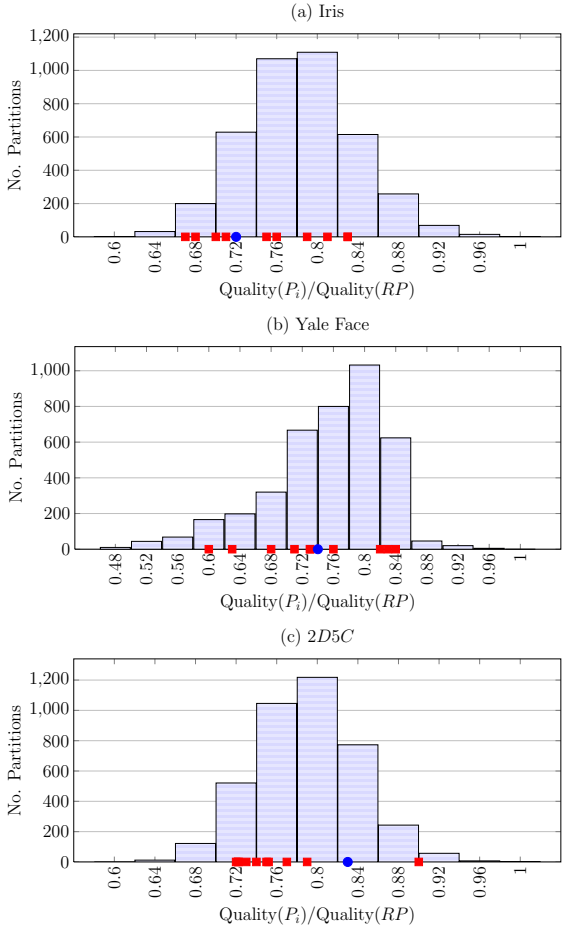


Fig. 3. Quality of Partitions.

References

1. M. Ackerman and S. Ben-David. Measures of Clustering Quality: A Working Set of Axioms for Clustering. *In proceedings of the* 22^{nd} *Neural Information Processing Systems*, 2008.
2. M. Ackerman and S. Ben-David. Clusterability: A theoretical study. *Journal of Machine Learning Research - Proceedings Track*, 5:1–8, 2009.
3. A. Agarwal, J. M. Phillips, and S. Venkatasubramanian. Universal multi-dimensional scaling. *In proceedings of the* 16^{th} *ACM SIGKDD*, 2010.
4. J. Aldrich. R. A. Fisher and the making of maximum likelihood 1912–1922. *Statist. Sci.*, 12(3):162–176, 1997.
5. E. Bae and J. Bailey. Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. *In proceedings of the* 6^{th} *ICDM*, 2006.
6. E. Bae, J. Bailey, and G. Dong. A clustering comparison measure using density profiles and its application to the discovery of alternate clusterings. *Data Mining and Knowledge Discovery*, 2010.
7. A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. In *Pacific Symposium on Biocomputing*, 2002.

(a) Iris (b) Yale Face (c) $2D5C$

Fig. 4. MDS rendering of the LiftEMD distances between all representative partitions.

Fig. 5. Visual illustration of two interesting representative partitions on Yale Face.

8. P. Berkhin. A survey of clustering data mining techniques. In J. Kogan, C. Nicholas, and M. Teboulle, editors, *Grouping Multidimensional Data*, pages 25–71. Springer, 2006.
9. R. E. Bonner. On some clustering techniques. *IBM J. Res. Dev.*, 8:22–32, January 1964.
10. R. Caruana, M. Elhawary, N. Nguyen, and C. Smith. Meta clustering. *Data Mining, IEEE International Conference on*, 0:107–118, 2006.
11. M. Coen, H. Ansari, and N. Fillmore. Comparing clusterings in space. In *ICML*, 2010.
12. X. H. Dang and J. Bailey. Generation of alternative clusterings using the CAMI approach. *In proceedings of* 10^{th} *SDM*, 2010.
13. X. H. Dang and J. Bailey. A hierarchical information theoretic technique for the discovery of non linear alternative clusterings. *In proceedings of the* 16^{th} *ACM SIGKDD*, 2010.
14. S. Das, A. Abraham, and A. Konar. *Metaheuristic Clustering*. Springer Publishing Company, Incorporated, 1st edition, 2009.
15. R. N. Dave. Validating fuzzy partitions obtained through c-shells clustering. *Pattern Recogn. Lett.*, 17:613–623, May 1996.
16. I. Davidson and Z. Qi. Finding alternative clusterings using constraints. *In proceedings of the* 8^{th} *ICDM*, 2008.
17. J. C. Dunn. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 1974.
18. A. Frank and A. Asuncion. UCI machine learning repository, 2010.
19. A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.
20. C. R. Givens and R. M. Shortt. A class of wasserstein metrics for probability distributions. *Michigan Math Journal*, 31:231–240, 1984.
21. D. Gondek and T. Hofmann. Non-redundant data clustering. *In proceedings of the* 4^{th} *ICDM*, 2004.

22. T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.
23. M. Halkidi and M. Vazirgiannis. Clustering validity assessment: finding the optimal partitioning of a data set. *In proceedings of the* 1^{st} *ICDM*, 2001.
24. M. Halkidi, M. Vazirgiannis, and Y. Batistakis. Quality scheme assessment in the clustering process. *In proceedings of the* 4^{th} *PKDD*, 2000.
25. W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.
26. P. D. Hoff. *A First Course in Bayesian Statistical Methods.* Springer, 2009.
27. A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*, 2010.
28. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31:264–323, September 1999.
29. P. Jain, R. Meka, and I. S. Dhillon. Simultaneous unsupervised learning of disparate clusterings. *Stat. Anal. Data Min.*, 1:195–210, November 2008.
30. S. Joshi, R. V. Kommaraju, J. M. Phillips, and S. Venkatasubramanian. Comparing distributions and shapes using the kernel distance (to appear). 27^{th} *Annual Symposium on Computational Geometry*, 2011.
31. G. G. C. Ma; and J. Wu. *Data Clustering: Theory, Algorithms, and Applications.* SIAM, Society for Industrial and Applied Mathematics, illustrated edition, May 2007.
32. D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, New York, NY, USA, 2002.
33. M. Meilă. Comparing clusterings–an information based distance. *J. Multivar. Anal.*, 2007.
34. N. Metropolis, A. W. Rosentbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 1953.
35. P. Michaud. Clustering techniques. *Future Generation Computer Systems*, 1997.
36. G. Milligan and M. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–179, 1985. 10.1007/BF02294245.
37. D. Niu, J. G. Dy, and M. I. Jordan. Multiple non-redundant spectral clustering views. In *ICML'10*, pages 831–838, 2010.
38. Z. Qi and I. Davidson. A principled and flexible framework for finding alternative clusterings. *In proceedings of the* 15^{th} *ACM SIGKDD*, 2009.
39. P. Raman, J. M. Phillips, and S. Venkatasubramanian. Spatially-aware comparison and consensus for clusterings (to appear). *Proceedings of* 11^{th} *SDM*, 2011.
40. W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.
41. G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk metropolis algorithms. *Annals of Applied Probability*, 7:110–120, 1997.
42. A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *JMLR*, 3:583–617, 2003.
43. P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition).* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.
44. S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Elsevier, 2006.
45. R. Xu and D. Wunsch. *Clustering.* Wiley-IEEE Press, 2009.
46. Y. Zhang and T. Li. Consensus clustering + meta clustering = multiple consensus clustering. *Florida Artificial Intelligence Research Society Conference*, 2011.
47. Y. Zhang and T. Li. Extending consensus clustering to explore multiple clustering views. *In proceedings of* 11^{th} *SDM*, 2011.
48. D. Zhou, J. Li, and H. Zha. A new Mallows distance based metric for comparing clusterings. In *ICML*, 2005.

Author Index