A Sampling-based Approach for Discovering Subspace Clusters

Sandy Moens[1], Boris Cule[1✉], and Bart Goethals[1,2]

[1] University of Antwerp, Belgium {firstname.lastname}@uantwerpen.be
[2] Monash University, Australia

Abstract. Subspace clustering aims to discover clusters in projections of highly dimensional numerical data. In this paper, we focus on discovering small collections of interesting subspace clusters that do not try to cluster all data points, leaving noisy data points unclustered. To this end, we propose a randomised method that first converts the highly dimensional database to a binarised one using projected samples of the original database. This database is then mined for frequent itemsets, which we show can be translated back to subspace clusters. In our extensive experimental analysis, we show on synthetic as well as real world data that our method is capable of discovering highly interesting subspace clusters.

1 Introduction

The main task of clustering is to group similar objects together, while keeping sufficiently different objects apart. However, due to the *curse of dimensionality*, traditional clustering methods struggle with high-dimensional data. In short, with high-dimensional data, the distances between pairs of objects, measured over all dimensions, become increasingly similar. As a result, no proper clusters can be formed, as all objects end up almost equally distant from each other.

Subspace clustering attempts to solve this problem by discovering clusters of objects that are similar in a limited number of dimensions. However, given the exponential complexity of the search space, identifying the relevant set of dimensions is computationally demanding, which is why existing subspace clustering methods suffer from long run-times [1]. Furthermore, some existing approaches produce full clusterings, thereby ensuring that each object is assigned to exactly one cluster. This is not always desirable: 1) the data may contain a lot of noise, that should ideally not be assigned to any cluster and 2) there is no reason why a particular object should not be assigned to multiple clusters, especially if the sets of dimensions that define these clusters are entirely different.

In this paper, we take a similar approach as CartiClus [2]: we first convert a numeric database to a transactional one and then use frequent pattern mining to extract subspace clusters. Our method can efficiently produce highly interesting subspace clusters, along with the dimensions that define them. We avoid the computational complexity of existing methods by deploying a randomised algorithm. We first take a large number of samples from the original data, such that

each sample consists of a number of objects in a fixed (random) set of dimensions. In each sample, we then cluster the objects, and subsequently assign all objects in the original data to the nearest cluster centroid. This produces a set of objects per centroid, which we interpret as a transaction. By merging the transactions produced for all different samples, we obtain a transaction database. We then sample maximal frequent itemsets from this database to obtain potential clusters. Finally, we identify the relevant dimensions for each discovered cluster.

The main contributions of this paper can be summarised as follows: we propose a randomised sampling algorithm that efficiently identifies localised clusters and their relevant dimensions, we allow data objects to be part of multiple clusters, and we leave noise objects unclustered, and we perform a theoretical evaluation to show the efficiency of our method and an extensive experimental evaluation to demonstrate the quality of our output.

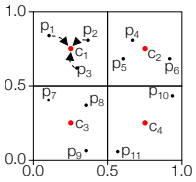
2 Background

Subspace clustering Let $\mathcal{D} = \{D_1, \ldots, D_m\}$ be a set of m dimensions. Each dimension D_i comes with a domain $dom(D_i)$. An m-dimensional data point $p = (d_1, \ldots, d_m)$ is a tuple of values over $\mathcal{D}$, such that $d_i \in dom(D_i)$ for each $i = \{1, \ldots, m\}$. The input database $\mathcal{P} = (p_1, \ldots, p_q)$ contains a collection of q such m-dimensional data points. Furthermore, each dimension D_i comes with a distance function $\delta_{D_i} : dom(D_i) \times dom(D_i) \rightarrow \mathbb{R}$. Additionally, we assume that for any subset of dimensions $\mathtt{D} = \{\mathtt{D}_1, \ldots, \mathtt{D}_l\}$, with $1 \leq l \leq m$ and $\mathtt{D} \subseteq \mathcal{D}$ there exists a distance function $\delta_{\mathtt{D}} : (dom(\mathtt{D}_1) \times \ldots \times dom(\mathtt{D}_l)) \times (dom(\mathtt{D}_1) \times \ldots \times dom(\mathtt{D}_l)) \rightarrow \mathbb{R}$. All used distance functions must satisfy the usual conditions (non-negativity, identity, symmetry and the triangle inequality). Given a subset of dimensions $\mathtt{D} \subseteq \mathcal{D}$, we denote by $p^{\mathtt{D}}$ a data point, and by $\mathtt{P}^{\mathtt{D}}$ a set of data points, projected onto the given dimensions. A *subspace cluster* S is a tuple containing a subset of datapoints and dimensions, i.e., $S = (\mathtt{P}, \mathtt{D})$, with $\mathtt{P} \subseteq \mathcal{P}$ and $\mathtt{D} \subseteq \mathcal{D}$.

Frequent itemset mining Let $\mathcal{I} = (i_1, \ldots i_n)$ be a finite set of n items. A *transaction* t is a subset of items. We denote by $\mathcal{T} = (t_1, \ldots, t_o)$ a database of o transactions. An *itemset* $\mathtt{I}$ is also a subset of items. A transaction t is said to support an itemset $\mathtt{I}$ if $\mathtt{I} \subseteq t$. The set of all transactions that support an itemset is called the *cover* of that itemset, i.e., $cov(\mathtt{I}) = \{t \mid t \in \mathcal{T} \wedge \mathtt{I} \subseteq t\}$. The *support* of an itemset is the size of its cover, i.e., $sup(\mathtt{I}) = |cov(\mathtt{I})|$. Given a minimal support threshold $\sigma \geq 0$, an itemset $\mathtt{I}$ is considered *frequent* if its support is larger than or equal to σ, i.e., $sup(\mathtt{I}) \geq \sigma$. An itemset $\mathtt{I}$ is called *maximal* if there exists no superset of $\mathtt{I}$ that is also frequent with respect to σ. The anti-monotonic property of the support of itemsets guarantees that all subsets of a frequent itemset are also frequent.

3 Randomised Subspace Clusters

Existing methods for discovering subspace clusters from numeric data often focus on the complete raw dataset to compute subspace clusters using a bottom-



(a)

centroid id	items
c_1	1, 2, 3
c_2	4, 5, 6
c_3	7, 8, 9
c_4	10, 11

(b)

Fig. 1. (a) A fictitious example dataset with 2 dimensions, 11 data points (black dots) and 4 centroids (red circles). (b) Binarised dataset in short format for the toy dataset.

up [3,1] or a top-down approach [4]. In this paper we introduce RASCL, which uses randomised subsets of the data (both in the data points and in the dimensions) as a starting point for detecting subspace clusters. The discovered clusters are then checked for occurrence in multiple subsamples of the data. If a cluster occurs frequently enough in the set of samples we output it as a subspace cluster. Our algorithm relies on two simple premises: 1) higher dimensional subspace clusters also form subspace clusters in lower dimensions; 2) if we take enough samples and use them to detect clusters, a lot of similar subclusters of the same true cluster will be found in different projections. Moreover, by repeating such a randomised procedure many times we end up with a stable solution.

3.1 Randomised data transformation

Data binarisation To binarise a numeric database $\mathcal{P}$ into a transaction database $\mathcal{T}$ we use the indices of data points as the items for $\mathcal{T}$, resulting in $|\mathcal{P}|$ items. In addition, we obtain a mapping between data points and items. Ideally, a transaction contains data points that are close together in some set of dimensions. Then an itemset (essentially a set of data points) that occurs in a large fraction of transactions can be seen as a subspace cluster over some set of dimensions.

We define a randomised process for constructing a single transaction database. We repeat this process n times and concatenate all transactions into a single database $\mathcal{T}^*$. We first sample a small subset of data points $\mathtt{P}$ and a small subset of dimensions $\mathtt{D}$ (the sampling strategy is explained below). The data points are projected onto the subset of dimensions and used as input for the K-means clustering algorithm. The resulting cluster centroids are used to partition the original data points, assigning each data point to the closest centroid. As such, each centroid represents one transaction and its items are the data points assigned to it. Formally, for a set of centroids $C^{\mathtt{D}}$ the closest centroid for a projected data point $p^{\mathtt{D}}$ is given by $c^{p^{\mathtt{D}}} = \text{argmin}_{c^{\mathtt{D}} \in C^{\mathtt{D}}}(\delta_{\mathtt{D}}(p^{\mathtt{D}}, c^{\mathtt{D}}))$.

Example 1: *Fig. 1(a) shows a toy database of 11 data points in a 2D space. A red circle represents a synthetic cluster centroid and the surrounding square visually shows data points closest to that centroid. When constructing the binarised database, the index of a data point is added to the transaction of the nearest cluster centroid. The resulting transaction database is shown in Fig. 1(b).*

Generating data samples As mentioned previously, our binarisation strategy requires a sample of data points and a sample of dimensions. The main question

now is how we can bias the sampling procedure to obtain samples that will have a higher potential to contain cluster structures.

For the data points, we can sample k data points uniformly at random. By repeating this a large number of times, we expect each cluster to be represented by a sufficient number of data points in a high enough number of samples.

For the dimensions, a naive solution would be to sample uniformly at random a subset of dimensions of size x, with $1 \leq x \leq |\mathcal{D}|$. However, since the number of combinations larger than 2 can blow up, a random sample of dimensions will likely be too large to contain a meaningful cluster. Sampling just one dimension may result in discovering cluster structures that do not span multiple dimensions. Our empirical results (omitted due to space constraints) have shown that sampling 2 dimensions results in higher quality clusters. We apply weighted sampling to boost the probability of sampling dimensions that contain cluster structures. Similar to Moise et al. [1], we assume that uniformly distributed dimensions do not contain any cluster structure. As such, to detect non-uniformity of a dimension we create a histogram using the Freedman-Diaconis' rule [5] to compute an appropriate number of bins for the data. This rule is robust to outliers and does not assume data to be normally distributed. Let us denote by B^D the bins for a given dimension using the Freedman-Diaconis' rule and let $|b|$ denote the number of data points falling in bin b. We compute how many bins contain less than the number of expected data points under uniform data distribution. The unnormalised sampling potential $\mathcal{W}$ of a dimension is given by

$$\mathcal{W}(D) = \sqrt{\frac{|\{b \mid b \in B^D \wedge |b| \leq \frac{|\mathcal{P}|}{|B^D|}\}|}{|B^D|}}. \tag{1}$$

The resulting distribution favours dimensions with more cluster potential.

Time complexity The worst case complexity of our binarisation method is mostly dependent on K-means. However, we use only a small subset of data points, typically $|\mathtt{P}| \ll |\mathcal{P}|$, to compute cluster centroids. For this small subset the complexity for clustering is $\mathcal{O}(n \times (|\mathtt{P}| \times |\mathtt{D}| \times K \times i))$ with n the number of database samples and i the number of iterations. Generation of samples for both data points can be done in $\mathcal{O}(|\mathcal{P}|)$ and for dimensions can be done in $\mathcal{O}(|\mathcal{D}|)$. Assignment of data points to cluster centroids is done in a single sweep, i.e., $\mathcal{O}(K \times |\mathcal{P}|)$. The total time complexity for generating samples and binarising the database is $\mathcal{O}(K \times |\mathcal{P}| + |\mathcal{D}| + n \times (|\mathtt{P}| \times |\mathtt{D}| \times K \times i))$.

3.2 Extracting subspace clusters

We previously constructed a binarised database $\mathcal{T}^*$ by concatenating n binarised ones built using random samples of data points and dimensions. The premise is that transactions represent cluster centroids and their items are indices of data points in their close proximity for the set of dimensions. Since we generated n samples, we know that each index occurs n times within $\mathcal{T}^*$. If then a set of items occurs often together in the database, i.e., it is a frequent itemset with high support, then we know that in many sets of dimensions the same set of data points

occur in close proximity, which is exactly the objective for a subspace cluster. However, typically the number of frequent itemsets is huge (largely because all subsets of frequent itemsets are frequent). To alleviate this problem we use maximal itemsets and, more particularly, our algorithm samples μ maximal frequent itemsets from the binarised database. The resulting itemsets are the data points for subspace clusters. An effective method for sampling maximal frequent itemsets was introduced by Moens and Goethals [6]. It iteratively extends an itemset with new items, until the set is found to be maximal given a threshold τ and a monotonic quality measure (e.g., support).

After extracting a collection of data points, we have to discover the dimensions in which the data points form a cluster. In contrast to some existing methods [1,4], we do not require to go back to the data itself to check each dimension individually, since our binarisation process preserved some essential information that can guide us here. That is, our algorithm previously sampled collections of dimensions which can be reused to determine a valid subset of dimensions. We denote by $dims(t)$ a map that for a transaction returns its *linked dimensions*, i.e., the dimensions that were used for its construction in the binarisation process. For a maximal itemset $\mathtt{I}$ we can use the transactions in its cover to determine its *relevant dimensions*, i.e., the set containing all linked dimensions for transactions in $cov(\mathtt{I})$. Formally, $dims(\mathtt{I}) = \{d | d \in \mathcal{D} \wedge d \in dims(t) \wedge t \in cov(\mathtt{I})\}$. An itemset $\mathtt{I}$, mapped to the data points $\mathtt{P}$, forms together with its relevant dimensions the subspace cluster $S = (\mathtt{P}, dims(\mathtt{I}))$.

3.3 Selecting the best subspace clusters

After discovering a large number of subspace clusters (depending on parameter μ), we finally select a small collection of r clusters that can be deemed the most interesting subspace clusters. The number of data points that is present in the subspace cluster is an indication that the same set of data points are often related even in different subsets of dimensions (experiments omitted due to space constraints). In our method we will employ this heuristic (i.e., the larger the cluster, the better) for sorting discovered subspace clusters. Finally, to reduce redundancy in the cluster results, we sequentially evaluate each cluster and select those clusters that have less than 25% cluster overlap with previously selected ones. Note that when sorting clusters using the number of objects, this results in smaller clusters as r increases. Finally, we exclude very small clusters with less than 10 data points.

4 Experiments

In our experiments, we use synthetic data provided by Günnemann et al. [7]. The dataset characteristics are shown in Table 1. To measure the performance we use PRECISION and RECALL scores on object level, as well as their harmonic mean F1. Additionally we use their dimensionality aware counterparts which are indicated by the subscripts D (for scores about the dimensions) and SC (for

	#rows	#dimensions	#clusters	#objects/cluster	#dimensions/cluster
$\text{dbsizescale}_{\text{s1500}}$	1,595	20	10	166.3	14.0
$\text{dbsizescale}_{\text{s2500}}$	2,658	20	10	276.5	14.0
$\text{dbsizescale}_{\text{s3500}}$	3,722	20	10	385.8	14.0
$\text{dbsizescale}_{\text{s4500}}$	4,785	20	10	496.2	14.0
$\text{dbsizescale}_{\text{s5500}}$	5,848	20	10	608.5	14.0
$\text{dimscale}_{\text{d05}}$	1,595	5	10	182.6	3.5
$\text{dimscale}_{\text{d10}}$	1,595	10	10	181.5	6.7
$\text{dimscale}_{\text{d25}}$	1,595	25	10	180.9	16.9
$\text{dimscale}_{\text{d50}}$	1,595	50	10	181.6	33.5
$\text{dimscale}_{\text{d75}}$	1,595	75	10	181.9	50.4
$\text{noisescale}_{\text{n10}}$	1,611	20	10	166.5	14.6
$\text{noisescale}_{\text{n30}}$	2,071	20	10	166.1	14.6
$\text{noisescale}_{\text{n50}}$	2,900	20	10	166.3	14.6
$\text{noisescale}_{\text{n70}}$	4,833	20	10	166.8	14.6

Table 1. Main characteristics of the synthetic datasets.

scores about the combination of objects and dimensions) [7]. Finally, we also use M^{E4SC} [7], a measure to assess the quality of clusterings. Note that unless stated otherwise, we assign just one discovered cluster to each ground truth cluster.

We compare two variants[1]: RASCL sets $k > K$ and RASCL_R sets $k = K$ which essentially skips the clustering step. For each dataset we use the ground truth and select the ground truth cluster with the largest overlap with the cluster being evaluated to compute its quality. We run each experiment 10 times and report the average results for the first r subspace clusters. Less than r clusters may be reported. Unless stated otherwise, we fix the following parameters: $n = 1000$, $k = 100$, $K = 20$, $\sigma = 200$, $\mu = 100$ and $r = 10$. We provide the following guidance: n should be set high enough to obtain a representative sample, k should be sufficiently larger than K for the clustering to make sense, r should be set to the desired number of clusters, μ should be high enough so no information is lost due to randomisation. K and σ are more difficult to set, but we show that the performance of RASCL is not overly sensitive to changes in their values.

Cluster quality We compare our methods to CARTICLUS [2] and PROCLUS [4]. We used different instantiations of RASCL and RASCL_R by varying K and σ. For CARTICLUS we use the parameter settings as selected by the authors [2] as basis for this experiment. For PROCLUS we set parameters following the ground truth.

Object quality results are shown in Fig. 2 (a-f). All algorithms perform very well with respect to PRECISION except PROCLUS, and setting $K = 20$ and $\sigma = 200$ we slightly outperform CARTICLUS. $\text{RASCL}_\text{R}^{K10\sigma100}$, $\text{RASCL}^{K10\sigma100}$ and $\text{RASCL}^{K20\sigma200}$ outperform the competitors on RECALL, while $\text{RASCL}_\text{R}^{K20\sigma200}$ often fails to deliver good results. This is due to the introduced randomness: using random centroids leads to more partially similar transactions. Combined with a high support this results in small subclusters of the true ground truth clusters.

Results for the dimension quality are shown in Fig. 2 (g-l). We see that our algorithms generally outperform the competitors by quite a margin and we see that our simple solution of using linked dimensions (Section 3.2) works really well. For smaller subspace clusters with lower PRECISION, the dimension quality

[1] Source code and experiments are available via `https://gitlab.com/adrem/rascl`

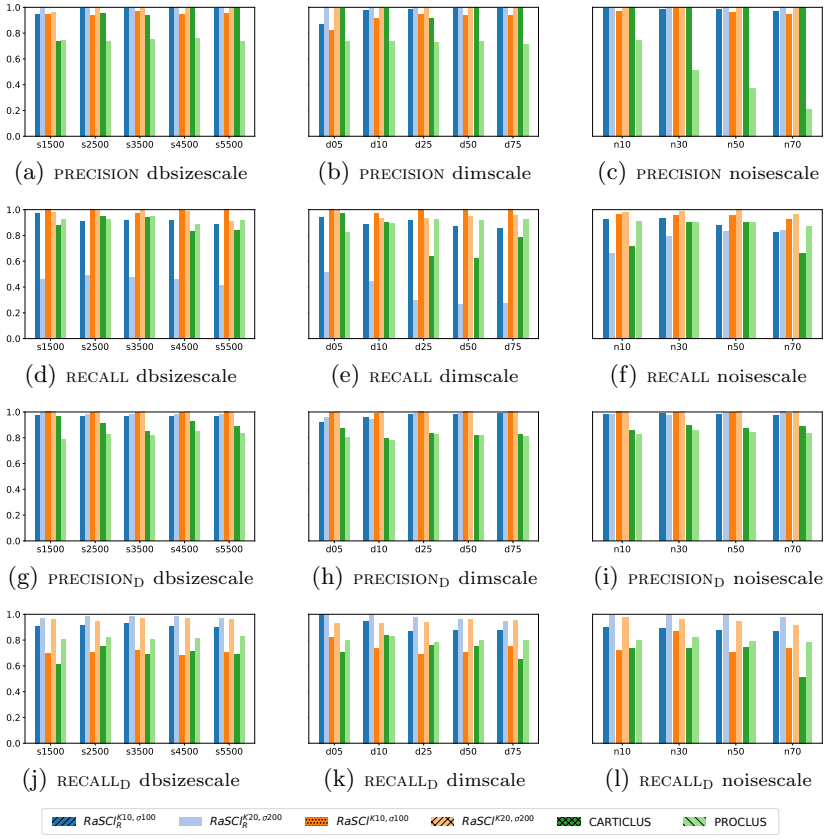
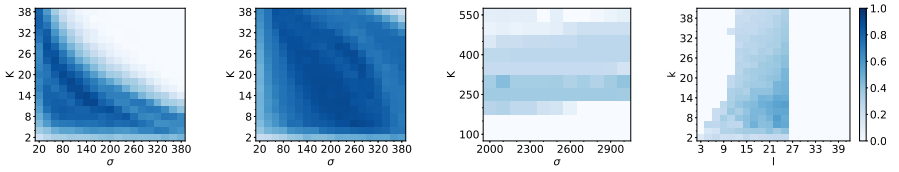


Fig. 2. Object quality (a-f) and dimension quality (g-l) scores for different datasets.

decreases as a result. Comparing ($K = 10$, $\sigma = 100$) to ($K = 20$, $\sigma = 200$), the latter produces better results, mostly because there are more linked dimensions tied to the cover of the maximal itemset, boosting RECALL$_D$.

Parameter sensitivity We test the influence of K and σ on the dimscale$_{d25}$ dataset and show the M^{E4SC} scores. Note that even though our algorithm is not meant for producing full clusterings, it produces very high quality results on this metric. Fig. 3 shows scores for the first 10 subspace clusters using various parameter settings for the algorithms. For our algorithm we use a window around the default parameters. For CARTICLUS we use a grid around the optimal parameters and for PROCLUS we define sensible grids. We see that RASCL is not overly susceptible to parameter changes and that, in general, the default parameters produce good and stable results. In contrast, RASCL$_R$ can still produce very good results, but the quality diminishes quickly when the parameters are not too far from the optimal parameters. Increasing K or σ results in subclusters of the true clusters, thus decreasing the overall score. We see that finding good settings for CARTICLUS and PROCLUS is much harder. For PROCLUS l cannot



(a) RASCL$_R$ d25 (b) RASCL d25 (c) CARTICLUS d25 (d) PROCLUS d25

Fig. 3. Grid search quality results for various methods on the dimscale$_{d25}$ dataset.

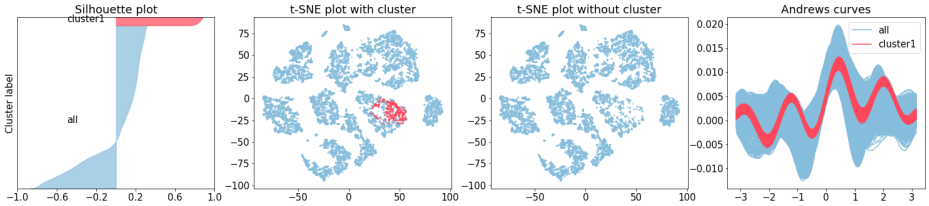


Fig. 4. Subspace cluster 1 for the pendigits datasets using RASCL.

exceed the number of dimensions in the data, resulting in lots of 0 scores in the figures. The experiments on other datasets produced similar results.

Real world datasets We tested our method on the pendigits dataset, a classification dataset found in the UCI machine learning repository[2]. Using RASCL with $n = 1000$, $k = 100$, $K = 10$, $\sigma = 100$, $\mu = 100$ and $r = 10$ we discover multiple subspace clusters for each class. A general trend we found was that the discovered clusters have a very high PRECISION of approx. 91%, but they have rather low RECALL averaging around 20%. We evaluate the largest subspace cluster in Fig. 4. The silhouette plot shows high similarity for points in the cluster (red) and a much lower score for points outside the cluster (blue). A similar trend is found in the scatter plots, which are obtained using *t-SNE* transformation [8] based on the relevant dimensions. The left scatter plot shows all data (blue) together with the subspace cluster (red), while the right scatter plot shows only data points not in the cluster. This shows that using our method we do not miss many data points that are within the region of the subspace cluster according to this transformation. The plot on the right is the Andrews plot [9], which is a smoothed parallel coordinates plot, showing cluster structures more clearly. Similar plots are found for the remaining clusters.

5 Related Work

Subspace clustering attempts to find clusters in subsets of dimensions. However, some traditional clustering models, unsuited to this setting, have been adapted for this purpose. PROCLUS [4], one of the first methods for subspace clustering, adapts K-means [10] to this setting. The analogy to our work is the initialisation:

[2] `https://archive.ics.uci.edu/ml/datasets.html`

a two-step randomised procedure is used to obtain an approximation to a *piercing set*, i.e., a set of points each from a different cluster, which are refined to clusters.

DOC [11] is an algorithm that finds subspace clusters using a Monte Carlo method to sample a random point from a cluster as well as a *discriminating* set of points. It then extends the random point to a full subspace cluster using a bounding box around that point. Its extension MineClus [12] uses the same medoid points for expanding the cluster, but it drops the randomised procedure. Similar to our approach, it also converts the data to a binarised dataset. Other clustering algorithms, such as DBSCAN [13], have also been adapted for the subspace clustering task [14]. Recently, more general techniques have been proposed for searching the subspace [15], where the discovery of clusters is left to specialised algorithms. However, all of the above methods are computationally very expensive as they search in an exponential set of subspaces.

FIRES [16] is a generic framework for finding subspace clusters, employing existing clustering techniques to compute a set of base clusters in single dimensions. These base clusters are then merged based on their similarity, and the resulting clusters are then pruned and refined to optimise accuracy. The CartiClus algorithm [2], like our method, creates a binarised dataset. However, in CartiClus, the dimensions are defined during the construction of transactions (or *carts*), such that all carts rely on the same dimension sets. Finally, the carts are mined for frequent itemsets which are then translated back to subspace clusters. Bi-clustering [17] also simultaneously clusters rows and columns of numeric matrices. However, bi-clusters allow for more general clusters as they, for instance, group rows with constant values for a set of columns or group columns that decrease similarly over a set of rows. Typically, such methods are used for analysis of biological data such as gene expression data.

6 Conclusion

In this paper, we present a novel method for discovering interesting clusters in high-dimensional data. We started by converting the original data into a transaction database by selecting a small number of random data objects, projecting them to a small number of random dimensions, then clustering them, and, finally, building transactions by assigning all data objects to their closest cluster centroids. We repeat this procedure many times and merge the results. We then sample maximal itemsets randomly from the resulting transaction database, and consider each such itemset to be a potentially interesting cluster of objects. Finally, for each discovered cluster, we identify a relevant set of dimensions.

A major advantage of our method is that, by using the two randomised procedures, we avoid both the combinatorial explosion of possible dimension sets, and the computational cost of frequent itemset mining. In addition, we do not attempt to produce full clusterings, and we allow data objects to be part of multiple clusters, while noise objects will not be part of any cluster at all. Experimentally, we demonstrate that our method produces quality clusters and

is not overly sensitive to changes in the parameter settings, which is crucial for an unsupervised learning task.

References

1. G. Moise, J. Sander, and M. Ester, "P3c: A robust projected clustering algorithm," in *Sixth Int. Conf. on Data Mining (ICDM'06)*. IEEE, 2006, pp. 414–425.
2. E. Aksehirli, B. Goethals, E. Muller, and J. Vreeken, "Cartification: A neighborhood preserving transformation for mining high dimensional data," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 937–942.
3. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, *Automatic subspace clustering of high dimensional data for data mining applications*. ACM, 1998, vol. 27, no. 2.
4. C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," in *ACM SIGMoD Record*, vol. 28, no. 2. ACM, 1999, pp. 61–72.
5. D. Freedman and P. Diaconis, "On the histogram as a density estimator: L 2 theory," *Probability theory and related fields*, vol. 57, no. 4, pp. 453–476, 1981.
6. S. Moens and B. Goethals, "Randomly sampling maximal itemsets," in *KDD Workshop on Interactive Data Exploration and Analytics*. ACM, 2013, pp. 79–86.
7. S. Günnemann, I. Färber, E. Müller, I. Assent, and T. Seidl, "External evaluation measures for subspace clustering," in *Proc of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 1363–1372.
8. L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
9. D. F. Andrews, "Plots of high-dimensional data," *Biometrics*, pp. 125–136, 1972.
10. J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.
11. C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. Murali, "A monte carlo algorithm for fast projective clustering," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM, 2002, pp. 418–427.
12. M. L. Yiu and N. Mamoulis, "Frequent-pattern based iterative projected clustering," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 689–692.
13. M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.
14. K. Kailing, H.-P. Kriegel, and P. Kröger, "Density-connected subspace clustering for high-dimensional data," in *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM, 2004, pp. 246–256.
15. H. V. Nguyen, E. Müller, J. Vreeken, F. Keller, and K. Böhm, "Cmi: An information-theoretic contrast measure for enhancing subspace cluster and outlier detection," in *SIAM Int. Conf. on Data Mining*. SIAM, 2013, pp. 198–206.
16. H.-P. Kriegel, P. Kroger, M. Renz, and S. Wurst, "A generic framework for efficient subspace clustering of high-dimensional data," in *Fifth IEEE international conference on data mining (ICDM'05)*. IEEE, 2005, pp. 8–pp.
17. S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: a survey," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 1, no. 1, pp. 24–45, 2004.