A Review on Projected Clustering Algorithms

Kevin Y. Yip[1] David W. Cheung[1] Michael K. Ng[2]

[1]Department of Computer Science and Information Systems
The University of Hong Kong, Pokfulam Road, Hong Kong.
Email: ylyip,dcheung@csis.hku.hk.

[2]Department of Mathematics
The University of Hong Kong, Pokfulam Road, Hong Kong.
Email: mng@maths.hku.hk.

Abstract: In this expository paper, we survey some of the latest developments on projected clustering in data mining.

AMS (MOS): Subject Classifications. 62H30.

1. INTRODUCTION

Given a large dataset, a common data mining task is to partition the records into different groups according to the attribute values, so that records in the same group share some common properties. If the correct grouping of the records is not known a priori, or this information is not used in grouping, the process will be called clustering. Formally, given a dataset with N records and d attributes (or dimensions, the two terms will be used interchangeably in this paper), a clustering algorithm partitions the records into k disjoint clusters so that each record falls into only one cluster (the notations N, d and k will be used throughout the current text). The objective of the algorithm is to maximize the similarity between records within the same cluster, and minimize that between records of different clusters. The similarity between two records is determined by a distance function or a similarity measure, such as Manhattan distance, Euclidean distance, and Pearson correlation.

The clustering problem has been studied extensively in different disciplines and many algorithms have been developed according to the specific domain requirements. Some recent ones include CLARANS [14], BIRCH [17], CURE [7], ROCK [8] and CHAMELEON [11]. The distance/similarity functions of these algorithms involve all attributes of the target datasets. This is appropriate if all or most attributes are relevant to every cluster, but may become less accurate if a significant amount of attributes is irrelevant to some clusters. The relevance of an attribute to a cluster is defined as how well an attribute is in distinguishing records of the cluster from other records.

As an illustration, suppose we have a dataset of medical records as shown in Table 1. The dataset contains 3 clusters, indicated by the "Disease A" column. Assume this column is not available to some researchers, and they want to partition the records into the 3 clusters. As seen from the table, the values of the age attribute are quite random, and so it is unlikely to be useful for the clustering task. If the researchers choose a

Person	Age	Virus Level	Blood Type	...	Disease A
1	35	0.6	AB		Uninfected
2	64	0.9	AB		Uninfected
3	27	1.1	AB		Uninfected
4	18	9.8	O		Infected
5	42	8.6	AB	...	Infected
6	53	11.3	B		Infected
7	37	0.7	O		Recovered
8	28	0.4	A		Recovered
9	65	0.9	B		Recovered

Table 1: A dataset with irrelevant attributes.

clustering algorithm that involves all attributes in the distance/similarity calculations, the age attribute will become a noise attribute and lower the clustering accuracy. Persons of similar ages will tend to be put into the same cluster, regardless of the actual clusters they should belong to.

Traditionally, the problem of irrelevant attributes is tackled by applying feature selection techniques (see for example [12]), which try to derive the most suitable set of features for the clustering task. This is a preprocessing step to clustering. Once the best features are determined, all distance/similarity calculations will be based on the whole set of new features. The process is effective in eliminating attributes that are irrelevant to all or most clusters, such as the age attribute in the previous example. However, in case each cluster has a different set of relevant attributes, some features will still be not useful for some clusters. In our example, the virus level attribute has a distinguishing high average value in the infected cluster, while the values in the other two clusters are low and similar. Obviously, the attribute is highly relevant to the infected cluster and less relevant to the other two. Similarly, the blood type attribute is more relevant to the uninfected cluster than the other two as there is a high density of value AB in the uninfected cluster. Ideally, a clustering algorithm should concentrate on the relevant attributes in distance/similarity calculations.

The projected clustering problem [1], or subspace clustering problem [3], is defined for such a situation. In addition to finding the member records of each cluster, a projected clustering algorithm also determines the set of relevant attributes for each cluster. There is no clear definition of how “high” the distinguishing power an attribute should have in order to be regarded as relevant to a cluster, but altogether the sets of relevant attributes should maximize the clustering accuracy. In the following discussion, the term “selected attributes” will be used to mean the attributes selected by a projected clustering algorithm for a cluster, which it treats as the ones most relevant to the cluster.

The main aim of this paper is to review existing projected clustering algorithms. The outline of the paper is as follows. In §2, we briefly discuss existing projected clustering algorithms. In §3, experimental results are reported to compare these projected clustering algorithms. Finally, some concluding remarks are given in §4.

2. PROJECTED CLUSTERING ALGORITHMS

There has been a number of different approaches to projected clustering.

- In CLIQUE [3] (and its successors such as ENCLUS [5] and MAFIA [13]), a cluster is defined as a set of adjacent hyper-rectangles each with a certain density of records. The algorithms try to identify one-dimensional clusters at the beginning, and repeatedly search for clusters with one more dimension until no more can be found.

  The algorithms effectively identify dense regions in all subspaces, but each record can fall into multiple regions. They are therefore not very suitable for applications that require disjoint clusters. Another limitation is their exponential time complexity with respect to the dimensionality of the clusters, which makes them not efficient in identifying high-dimensional clusters.

- Another approach is proposed in [9]. The algorithm constructs a hypergraph with each item as a vertex and each frequent itemset as a hyperedge. Each hyperedge is assigned a weight equal to a function (e.g. average) of the confidences of all the association rules between the connecting items. A hypergraph partitioning method (such as [10]) is then applied to divide the hypergraph into k clusters, so that the sum of the weights that straddle clusters is minimized. Finally, each record is assigned to the cluster which rules are most compatible to it.

  The clusters formed by this algorithm are disjoint, but each item (attribute value) can only belong to a single cluster, which may be too restrictive. It also seems difficult to apply the algorithm on datasets with numeric attributes.

- A recent study [15] proposes a Monte Carlo algorithm called DOC to find projected clusters defined as hypercubes of width w. To find a cluster, a pivot point is randomly chosen as its center, and a small record set is randomly sampled to select its relevant dimensions. A dimension is selected if and only if all sample points are within a projected distance w from the pivot point along the dimension. All records in the dataset that fall into the defined cube will become a member of the cluster. The process repeats for different random samples and pivot points, and the cluster that maximizes a function of the number of selected dimensions and the number of records is returned. The whole process can then repeat to find other clusters.

  Since all records are examined after finding each set of dimensions, the execution speed can be quite low. A modified version called FastDOC is therefore proposed in the same paper. For each pivot point, the random samples are only used to find the relevant dimensions. Only the best set of dimensions (that has the largest size, or with a size exceeding a threshold) will be used to find the member records. The dataset is therefore scanned only once for each pivot point. This reduces the execution time significantly, but the probability of getting correct clusters is no longer guaranteed.

Both DOC and FastDOC have a clean and simple design, yet they appear to have some limitations. It seems quite unrealistic to assume clusters with equal width along all relevant dimensions. The parameter values (such as w) are difficult to determine, and the resulting clusters are only 2-approximations with width $2w$. In addition, according to the algorithm of DOC, clustering a small dataset with 20 dimensions using default parameter values requires more than 10 million iterations for each pivot point. On the other hand, the clusters formed by FastDOC tend to have too few records as the resulting cluster size is not considered when evaluating the sets of selected dimensions.

- Among the various approaches, PROCLUS [1] is one of the most efficient algorithms. It is a partitional clustering algorithm based on k-medoids. At the first iteration, k medoids are drawn randomly and each of them is assigned a set of "neighbors", which are records closer to it then its nearest medoid. The neighbors of a medoid are used to score the d dimensions of it, such that the shorter is the average distance of the neighbors from it along a dimension, the better score the dimension will receive. After calculating all $k \times d$ scores for all medoids, a total of $k \times l$ dimensions with the best scores will be selected (l is a user-parameter representing the average number of relevant dimensions per cluster), with a constraint that each cluster must select at least two dimensions. Each record in the dataset is then assigned to the cluster with the medoid closest to it, where the distance calculations only involve, and are normalized by, the selected dimensions. After record assignment, a clustering score will be calculated for the current set of clusters based on the normalized average intra-cluster distance from centroid. Clusters with too few records will be discarded and the medoids replaced by some other records, and a new iteration starts. If the new iteration does not give a better score, the old set of medoids will be restored, small clusters of it discarded, and a new iteration starts. The clustering process will stop when the best set of medoids remains unchanged for a number of iterations. Finally, records too far away from their medoids will be regarded as outliers.

  PROCLUS enjoys the inherent advantages of partitional clustering algorithms: efficient, low memory requirement, and guaranteed to form k clusters. A limitation of PROCLUS is its dependency on the input l, the average number of relevant dimensions. When a wrong l is specified, the accuracy of PROCLUS can deteriorate seriously. In real datasets, it is seldom possible to know the correct value of l. It is therefore difficult to apply PROCLUS on datasets with many (e.g. > 1000)

Attribute \ Record set	A_1	A_2	A_3	A_4
Whole dataset (mean/variance)	5.0/0.2	4.9/0.3	7.3/5.0	7.2/6.5
Cluster C_1 (medoid/average difference from medoid)	4.9/0.1	5.1/0.2	2.8/0.5	3.1/0.6

Table 2: The situation where global statistics is necessary in attribute selection.

dimensions.

We also discover that the dimension selection procedure of PROCLUS can be biased by "virtual compactness" of attribute values as illustrated by Table 2.

The table shows the statistics of a dataset and one of its clusters, C_1. In C_1, A_1 and A_2 have lower average distances from medoid than A_3 and A_4, therefore PROCLUS will give better scores to A_1 and A_2. However, if we consider the statistics of the whole dataset (what we call the "global statistics"), A_1 and A_2 are hardly useful for clustering because all records in the dataset have similar values. A_3 and A_4, on the other hand, are more powerful in identifying records that belong to C_1, i.e. of higher relevance to it.

- An improved version of PROCLUS is proposed in [2]. The algorithm, called ORCLUS, creates generalized clusters in arbitrary subspaces. It repeats 3 operations in successive iterations: record assignment, vector finding, and cluster merging. In the vector finding step, each cluster computes its covariance matrix and selects the eigenvectors with the smallest eigenvalues. This ensures low variance (i.e. high density) along the vectors. In the record assignment step of the next iteration, the distance between a record and a medoid (the centroid of a previous cluster) will be calculated in the space spanned by the selected vectors. The numbers of medoids and eigenvectors to pick in the first iteration are user parameters. The values are decreased by constant factors after each iteration, so that they will reach their final (user-specified) values at the same time.

  Having the ability to form clusters in arbitrary subspaces, ORCLUS is more robust than PROCLUS in handling datasets with dependent attributes. Also, by forming a larger number of clusters at the beginning, the initial error due to random seeding is reduced. A potential limitation of ORCLUS is the need to input the final number of vectors to select, which is hardly known to users. Although ORCLUS provides a way to evaluate this input value, it is only a post-clustering evaluation mechanism and the best number to use is not determined automatically from data. Restricting each cluster to have the same number of selected vectors is also a potential problem of ORCLUS, since the variance of attribute values along the vectors can vary across different clusters. In addition, since the vector selection process does not make use of global statistics, the problem illustrated in Table 2 can happen. Although ORCLUS mentions the use of global statistics in evaluating the projected dimensionality, it is not used in the clustering algorithm.

According to our experiment results, ORCLUS can also fail in cases where the dimensionality of the clusters is small compared to the dataset dimensionality, or when the number of records is small. Since record assignment in the first iteration is based on all the original dimensions, the former case will lead to the involvement of many noise attributes. The latter case is problematic as the covariance matrix of each cluster can reflect its data characteristics only if there are sufficient number of records in it. These problems limit the kinds of datasets that can be clustered by ORCLUS.

- A hierarchical clustering algorithm called HARP (a Hierarchical approach with Automatic Relevant attribute selection for Projected clustering) was proposed by Yip *et al.* [16]. Its main characteristic is its ability to automatically determine the relevant attributes of each cluster without requiring any hard-to-determine user parameters. It also defines some customizable procedures that can be implemented in different ways for different applications.

  The basic assumption of HARP is that if two records/sets of records are similar in a high dimensional space, they have a high probability of belonging to the same real cluster in a certain lower dimensional subspace. Based on this assumption, at any time, a cluster selects all and only attributes with a relevance up to a certain level, and two clusters are allowed to merge only if the resulting cluster can select a certain number of attributes. This approach has a number of advantages. (i) It provides a mechanism to automatically determine the number of selected attributes for each cluster. (ii) It ensures all attributes of high relevance are selected, and all selected attributes have a guaranteed relevance to the cluster. (iii) It prohibits non-similar records from being put into the same cluster, which is possible in the partitional approaches if each record is assigned to a cluster no matter how long is the distance between them.

  HARP has few restrictions on the dataset (e.g. can work on datasets of very high dimensionality) and the resulting clusters (e.g. each cluster can select different number of attributes). In order to handle datasets with many attributes but few records, HARP does not perform vector transformation and assumes all clusters are defined in subspaces orthogonal to the original dimensions.

3. EXPERIMENTAL RESULTS IN ACCURACY

Table 3 shows the datasets used in our accuracy experiments. The synthetic datasets are divided into two groups ***1 and ***2. The ***2 group has a smaller average number

Name	Type	Rec.	Class	Cat./Num. Attribute	Avg. Rel. Attribute
Cat1	Syn.	500	5	20/ 0	12
Mix1	Syn.	500	5	10/ 10	12
Num1	Syn.	500	5	0/ 20	12
Cat2	Syn.	500	5	20/ 0	7
Mix2	Syn.	500	5	10/ 10	7
Num2	Syn.	500	5	0/ 20	7

Table 3: Datasets for the accuracy experiments.

of relevant attributes, which makes the advantage of projected clustering more obvious. Each of the 6 synthetic datasets contains 5% of artificial outliers.

We compared the results of HARP, PROCLUS, a traditional partitional clustering algorithm (simulated by PROCLUS with l set to d), ORCLUS, and FastDOC. We modified the distance function of PROCLUS so that it could handle categorical attributes. The distance between two records along an attribute is 0 if they have the same attribute value, or 1 otherwise. Since ORCLUS performs vector transformation, we only experimented it on numeric datasets. For synthetic datasets, we supplied the actual average number of relevant attributes to PROCLUS and ORCLUS. For all experiments on PROCLUS and ORCLUS, the same experiments were repeated 20 times.

Since DOC requires too many dataset scans, we did our experiments with FastDOC instead. Like ORCLUS, we experimented it on numeric datasets only. We set α to the fraction of records of the largest cluster, β to the default value of 0.25, w to 4 times the standard deviation of the relevant attribute with the largest variance (which is still 1.5 to 2 times smaller than the standard deviation of a random attribute), d_0 to the largest number of relevance attributes of the clusters, and MAXITER to 1000000. To create disjoint clusters, each time FastDOC reported a cluster, the records of it were removed. Originally we set k to the actual number of clusters, but the results showed that many non-outlier records were not put into any cluster. We therefore set k to 20 so that most non-outlier records could be clustered. Records not falling in any cluster are treated as outliers. The experiments were repeated 10 times.

The results of the synthetic datasets are shown in Table 4. For PROCLUS, ORCLUS and FastDOC, the results with the best scores are shown on the first line and the average results on the second line. The two numbers in each cell represent the percentage of error cases and records being discarded as outliers. If we denote $|C_i|_{class=j}$ as the number of records in cluster C_i with class label j, the error percentage is calculated by the formula

$$\frac{\sum_i(|C_i| - max_j |C_i|_{class=j})}{N} \times 100$$

Data	HARP.1	PRO-CLUS	Trad.	OR-CLUS	Fast-DOC‡
Cat1	0.0/5.0	3.6/1.4 6.7/3.7	2.6/26.4 5.8/5.3	N/A	N/A
Mix1	0.4/6.8	2.2/17.0 6.8/10.1	11.6/11.2 7.9/4.6	N/A	N/A
Num1	0.8/5.0	1.8/21.4 7.2/8.3	4.4/32.0 5.9/9.2	0.4/23.8 2.31/8.15	0.4/ 7.4 0.04/ 8.3
Cat2	4.0/8.4	11.0/31.0 25.0/14.4	17.8/23.8 28.5/5.4	N/A	N/A
Mix2	11.4/4.4	16.6/62.2 25.4/32.8	17.6/38.6 24.1/11.6	N/A	N/A
Num2	18.8/4.4	11.6/50.8 18.7/20.7	11.6/28.0 23.3/10.9	50.8/0† 57.2/0	16.2/ 12 15.2/10.8

† Outlier removal disabled ‡ 20 clusters were formed

Table 4: Error and outlier percentage on synthetic datasets.

From the table, HARP outperforms traditional clustering methods by having higher accuracy and/or fewer non-outliers being discarded. Comparing HARP with the projected clustering algorithms, PROCLUS tends to discard too many records as outliers and in general, it has lower accuracy. ORCLUS performs well on SynNum1 (which has 12 relevant attributes out of 20), but discards almost 100% of the records as outliers when clustering SynNum2 (which has 7 relevant attributes out of 20, the results are not shown). In view of this, we disabled the outlier removal option and repeated the experiments on SynNum2. The resulting accuracy is poor. We suggest that the unsatisfactory performance is due to the small number of relevant attributes and the insufficient number of records, which resulted in wrong initial record assignments and biased vector transformation. We believe the accuracy of the two algorithms would improve if their attribute/vector selection procedures are modified to make use of global statistics.

For FastDOC, even we have supplied the correct parameter values and k was set to 20, there were still too many outliers discarded. From these results, HARP may be more suitable for creating disjoint clusters with different variance along different relevant attributes.

4. CONCLUDING REMARKS

In this paper, we reviewed existing projected clustering algorithms. Experimental results on synthetic datasets showed that HARP has improved accuracy over some other projected clustering algorithms in general.

We remark that clustering is a popular data-mining tool for gene expression profiles. A common use is to group similar genes into clusters according to their expression patterns and observe if any unclassified genes are put into a cluster of genes with some common properties. Such observation can suggest potential properties of the unclassified genes and provide insights for further investigation [6]. However, few projected clustering results have been reported on real gene expression profiles. For the future work, we study the effectiveness of the existing projected clustering algorithms on gene expression data by comparing their performance with non-projected algorithms and examine whether the attribute selection methods are biologically intuitive.

References

[1] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *ACM SIGMOD International Conference on Management of Data*, 1999.

[2] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *ACM SIGMOD International Conference on Management of Data*, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD International Conference on Management of Data*, 1998.

[4] Y. Barash and N. Friedman. Context-specific bayesian clustering for gene expression data. In *Annual Conference on Research in Computational Molecular Biology*, 2001.

[5] C. H. Cheng, A. W.-C. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Knowledge Discovery and Data Mining*, pages 84–93, 1999.

[6] A. Gasch, P. Spellman, P. Brown and D. Botstein. Genomic expression programs in the respone of yeast cells to environmental changes. *Molecular Biology of the Cell*, pages 4241–4257, 2000.

[7] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *ACM SIGMOD International Conference on Management of Data*, 1998.

[8] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *15th International Conference on Data Engineering*, 1999.

[9] E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *1997 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[10] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. In *ACM/IEEE Design Automation Conference*, 1997.

[11] G. Karypis, E.-H. S. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[12] H. Liu and H. Motoda, editors. *Feature Extraction Construction and Selection: A Data Mining Perspective.* Kluwer Academic Publishers, 1998.

[13] H. Nagesh, S. Goil, and A. Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets, 1999.

[14] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings*, 1994.

[15] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *ACM SIGMOD International Conference on Management of Data*, 2002.

[16] K. Yip, D. Cheung and M. Ng. A highly-usable projected clustering algorithm. Technical report HKU CSIS.

[17] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *ACM SIGMOD International Conference on Management of Data*, 1996.