Subspace Clustering of High Dimensional Data

Carlotta Domeniconi
George Mason University
carlotta@ise.gmu.edu
Dimitris Papadopoulos   Dimitrios Gunopulos
University of California Riverside
{dimitris,dg}@cs.ucr.edu
Sheng Ma
IBM T. J. Watson Research Center
shengma@us.ibm.com

Abstract

Clustering suffers from the curse of dimensionality, and similarity functions that use all input features with equal relevance may not be effective. We introduce an algorithm that discovers clusters in subspaces spanned by different combinations of dimensions via local weightings of features. This approach avoids the risk of loss of information encountered in global dimensionality reduction techniques, and does not assume any data distribution model. Our method associates to each cluster a weight vector, whose values capture the relevance of features within the corresponding cluster. We experimentally demonstrate the gain in perfomance our method achieves, using both synthetic and real data sets. In particular, our results show the feasibility of the proposed technique to perform simultaneous clustering of genes and conditions in microarray data.

1 Introduction

The clustering problem concerns the discovery of homogeneous groups of data according to a certain similarity measure. Given a set of multi-dimensional data, clustering finds a partition of the points into clusters such that the points within a cluster are more similar to each other than to points in different clusters.

Clustering suffers from the curse of dimensionality. In high dimensional spaces, it is highly likely that, for any given pair of points within the same cluster, there exist at least a few dimensions on which the points are far apart from each other. As a consequence, distance functions that equally use all input features may not be effective. Furthermore, several clusters may exist in different subspaces, comprised of different combinations of features. In many real world problems, in fact, some points are correlated with respect to a given set of dimensions, and others are correlated with respect to different dimensions. Each dimension could be relevant to at least one of the clusters.

One solution to high dimensional settings consists in reducing the dimensionality of the input space. Traditional feature selection algorithms select certain dimensions in advance. Methods such as Principal Component Analysis (PCA) [9] transform the original input space into a lower dimensional space by constructing dimensions that are linear combinations of the given features, and are ordered by nonincreasing variance. While PCA may succeed in reducing the dimensionality, it has major drawbacks. The new dimensions can be difficult to interpret, making it hard to understand clusters in relation to the original space. Furthermore, all global dimensionality reduction techniques (like PCA) are not effective in identifying clusters that may exist in different subspaces. In this situation, in fact, since data across clusters manifest different correlations with features, it may not always be feasible to prune off too many dimensions without incurring a loss of crucial information. This is because each dimension could be relevant to at least one of the clusters.

These limitations of global dimensionality reduction techniques suggest that, to capture the local correlations of data, a proper feature selection procedure should operate locally in the input space. Local feature selection allows different distance measures to be embedded in different regions of the input space; such distance metrics reflect local correlations of data. In this paper we propose a *soft* feature selection procedure that assigns (local) weights to features according to the local correlations of data along each dimension. Dimensions along which data are loosely correlated receive a small weight, that has the effect of elongating distances along that dimension. Features that correlate strongly with data receive a large weight, which has the effect of constricting distances along that dimension.

2 Related Work

The problem of finding different clusters in different subspaces of the original input space has been addressed in [3]. The authors use a density based approach to identify clusters. The algorithm (CLIQUE) proceeds from lower to higher dimensionality subspaces and discovers dense regions in each subspace. While the work in [3] successfully introduces a methodology for looking at different subspaces for different clusters, it does not compute a partitioning of the data into disjoint groups. In fact, the reported dense regions largely overlap. On the other hand, for many applications such as customer segmentation and trend analysis, a partition of the data is desirable since it provides a clear interpretability of the results.

Recently [11], another density-based projective clustering algorithm (DOC/FastDOC) has been proposed. This approach requires the maximum distance between attribute values (i.e. maximum width of the bounding hypercubes) as parameter in input, and pursues an optimality criterion defined in terms of density of each cluster in its corresponding subspace. A Monte Carlo procedure is then developed to approximate with high probability an optimal projective cluster. In practice it may be difficult to set the parameters of DOC, as each relevant attribute can have a different local variance.

The problem of finding different clusters in different subspaces is also addressed in [1]. The proposed algorithm (PROjected CLUStering) seeks subsets of dimensions such that the points are closely clustered in the corresponding spanned subspaces. Both the number of clusters and the average number of dimensions per cluster are user-defined parameters. In contrast to the PROCLUS algorithm, our method does not require to specify the average number of dimensions to be kept per cluster. For each cluster, in fact, *all* features are taken into consideration, but properly weighted. The PROCLUS algorithm is more prone to loss of information if the number of dimensions is not properly chosen. ORCLUS [2] modifies the PROCLUS algorithm by adding a merging process of clusters, and selecting for each cluster principal components instead of attributes.

3 Problem Statement

We define what we call *weighted cluster*. Consider a set of points in some space of dimensionality N. A *weighted cluster* C is a subset of data points, together with a vector of weights $\mathbf{w} = (w_1, \ldots, w_N)$, such that the points in C are closely clustered according to the L_2 norm distance weighted using $\mathbf{w}$. The component w_j measures the degree of correlation of points in C along feature j. The problem becomes now how to estimate the weight vector $\mathbf{w}$ for each cluster in the data set.

In this setting, the concept of *cluster* is not based only on points, but also involves a weighted distance metric, i.e., clusters are discovered in spaces transformed by $\mathbf{w}$. Each cluster is associated with its own $\mathbf{w}$, that reflects the correlation of points in the cluster itself. The effect of $\mathbf{w}$ is to transform distances so that the associated cluster is reshaped into a dense hypersphere of points separated from other data. In traditional clustering, the partition of a set of points is induced by a set of *representative* vectors, also called *centroids*. The partition induced by discovering weighted clusters is formally defined as follows.

Definition: Given a set S of D points $\mathbf{x}$ in the N-dimensional Euclidean space, a set of k centers $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$, $\mathbf{c}_j \in \Re^N$, $j = 1, \ldots, k$, coupled with a set of corresponding weight vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$, $\mathbf{w}_j \in \Re^N$, $j = 1, \ldots, k$, partition S into k sets $\{S_1, \ldots, S_k\}$: $S_j = \{\mathbf{x} | (\sum_{i=1}^N w_{ji}(x_i - c_{ji})^2)^{1/2} < (\sum_{i=1}^N w_{li}(x_i - c_{li})^2)^{1/2}, \forall l \neq j\}$, where w_{ji} and c_{ji} represent the ith components of vectors $\mathbf{w}_j$ and $\mathbf{c}_j$ respectively (ties are broken randomly).

The set of centers and weights is *optimal* with respect to the Euclidean norm, if they minimize the error measure: $E_1(C, W) = \sum_{j=1}^k \sum_{i=1}^N w_{ji} e^{-X_{ji}}$ subject to the constraints $\sum_{i=1}^N w_{ji}^2 = 1 \; \forall j$. C and W are $(N \times k)$ matrices whose column vectors are $\mathbf{c}_j$ and $\mathbf{w}_j$ respectively, i.e. $C = [\mathbf{c}_1 \ldots \mathbf{c}_k]$ and $W = [\mathbf{w}_1 \ldots \mathbf{w}_k]$. X_{ji} represents the average distance from the centroid $\mathbf{c}_j$ of points in cluster j along dimension i, and is defined as $X_{ji} = \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2$, where $|S_j|$ is the cardinality of set S_j. The exponential function E_1 has the effect of making the weights w_{ji} more sensitive to changes in X_{ji}, and therefore to changes in local feature relevance. This allows larger error improvements as we adapt the values of weights and centers, and therefore a faster computation to achieve spherically shaped clusters (separated from each other) in the space transformed by optimal weights. In the following we present an algorithm that finds a solution (set of centers and weights) that is a local minimum of the error function E_1.

4 Locally Adaptive Clustering Algorithm

We start with *well-scattered* points in S as the k centroids: we choose the first centroid at random, and select the others so that they are far from one another, and from the first chosen center. We initially set all weights to $1/\sqrt{N}$. Given the initial centroids $\mathbf{c}_j$, for $j = 1, \ldots, k$, we compute the corresponding sets S_j as given in the definition above. We then compute the average distance along each dimension from the points in S_j to $\mathbf{c}_j$. Let X_{ji} denote this average distance along dimension i. The smaller X_{ji} is, the larger is the correlation of points along dimension i. We use the value X_{ji} in an exponential weighting scheme to credit weights to features (and to clusters): $w_{ji} = exp(-h \times X_{ji})/(\sum_{l=1}^{N}(exp(-h \times 2 \times X_{jl})))^{1/2}$, where h is a parameter that can be chosen to maximize (minimize) the influence of X_{ji} on w_{ji}. We empirically determine the value of h through cross-validation in our experiments with simulated data. We set the value of h to 9 in the experiments with real data (DNA microarray). The exponential weighting is more sensitive to changes in local feature relevance [5] and gives rise to better performance improvement. Note that the technique is centroid-based because weightings depend on the centroid. The computed weights are used to update the sets S_j, and therefore the centroids' coordinates. The procedure is iterated until convergence is reached, i.e. no change in centers' coordinates is observed. The resulting algorithm, that we call LAC (Locally Adaptive Clustering), is summarized in the following.

```
Input: $D$ points $\mathbf{x} \in R^N$, $k$, and $h$.
1. Start with $k$ initial centroids $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$;
2. Set $w_{ji} = 1/\sqrt{N}$, for each centroid $\mathbf{c}_j$, $j = 1, \ldots, k$ and each feature $i = 1, \ldots, N$;
3. For each centroid $\mathbf{c}_j$, and for each point $\mathbf{x}$:
    • Set $S_j = \{\mathbf{x} | j = arg\min_l L_w(\mathbf{c}_l, \mathbf{x})\}$, $L_w(\mathbf{c}_l, \mathbf{x}) = (\sum_{i=1}^{N} w_{li}(c_{li} - x_i)^2)^{1/2}$;
4. Compute new weights. For each centroid $\mathbf{c}_j$, and for each feature $i$:
    • Set $X_{ji} = \sum_{\mathbf{x} \in S_j}(c_{ji} - x_i)^2/|S_j|$;
    • Set $w_{ji} = exp(-h \times X_{ji})/(\sum_{l=1}^{N} exp(-h \times 2 \times X_{jl}))^{1/2}$;
5. For each centroid $\mathbf{c}_j$, and for each point $\mathbf{x}$:
    • Recompute $S_j = \{\mathbf{x} | j = arg\min_l L_w(\mathbf{c}_l, \mathbf{x})\}$;
6. Compute new centroids. Set $\mathbf{c}_j = \frac{\sum_{\mathbf{x}} \mathbf{x} 1_{S_j}(\mathbf{x})}{\sum_{\mathbf{x}} 1_{S_j}(\mathbf{x})}$, for each $j = 1, \ldots, k$, where $1_S(.)$ is the indicator function of set $S$;
7. Iterate 3,4,5,6 until convergence.
```

We prove the convergence of LAC in [7].

5 Experimental Evaluation

In our experiments we have designed 5 simulated data sets. Clusters are distributed as multivariate gaussians with different means and standard deviations. We have tested problems with two and three clusters up to 50 dimensions. For each problem, we have generated five or ten training sets, and for each of them an independent test set. We report accuracy results obtained via 5(10)-fold cross-validation comparing LAC, PROCLUS, DOC, and K-means. Among the subspace clustering techniques available in the literature, we chose PROCLUS [1] and DOC [11] because they compute a partition of the data. On the contrary, the CLIQUE technique [3] and the technique in [6] allow overlapping between clusters, and thus their results are not directly comparable with ours. Error rates for simulated data are computed according to the confusion matrices. The k centroids for the four algorithms are initialized by choosing well-scattered points among the given data. To facilitate the interpretation of weight values, we require that $\sum_i w_{ji} = 1 \; \forall j$ in our experiments, by properly adjusting the normalization factor of the weighting scheme.

Simulated Data

Example1: $N = 2$ and $k = 3$. All three clusters are distributed according to multivariate gaussians. Mean vector and standard deviations for each cluster are respectively: $(2, 0)$ and $(4, 1)$; $(10, 0)$ and $(1, 4)$; $(18, 0)$ and $(4, 1)$. We generated 60000 data points, and performed 10-fold cross-validation with 30000 training data and 30000 testing data. **Example2**: $N = 30$ and $k = 2$. Both clusters are distributed according to multivariate gaussians. Mean vector and standard deviations for each cluster are respectively: $(1, \ldots, 1)$ and $(10, 5, 10, 5, \ldots, 10, 5)$; $(2, 1, \ldots, 1)$ and $(5, 10, 5, 10, \ldots, 5, 10)$. We generated 10000 data points, and performed 10-fold cross-validation with 5000 training and 5000 testing data. **Example3**: $N = 50$ and $k = 2$. Both clusters are distributed according to multivariate gaussians. Mean vector and standard deviations for each cluster are: $(1, \ldots, 1)$ and $(20, 10, 20, 10, \ldots, 20, 10)$; $(2, 1, \ldots, 1)$ and $(10, 20, 10, 20, \ldots, 10, 20)$. We generated 10000 data points, and performed 10-fold cross-validation with 5000 training data and 5000 testing data. **Example 4**: This data set consists of off-axis oriented clusters, with $N = 2$ and $k = 2$. We generated 20000 data points, and performed 5-fold-cross-validation with 10000 training data and 10000 testing data. **Example 5**: This data set constists again of off-axis oriented two dimensional clusters. This data set contains three clusters. We generated 30000 data points, and performed 5-fold-cross-validation with 15000 training data and 15000 testing data.

Real Data

In our experiments we used seven different real data sets. The OQ-letter, Wisconsin breast cancer, Pima Indians Diabete, and Sonar data sets are taken from the UCI Machine Learning Repository. The Image data set is obtained from the MIT Media Lab. The characteristics of these data sets are as follows. **OQ**: 1536 records, $N = 16$, and $k = 2$; **Breast**: 683 records, $N = 9$, and $k = 2$; **Pima**: 768 records, $N = 8$, and

Table 1: Average error rates for simulated data.

	LAC	*PROCLUS*	*K-means*	*DOC*
Ex1	11.4±0.3	13.8±0.7	24.2±0.5	35.2± 2.2
Ex2	0.5±0.4	27.9±9.8	48.4±1.1	*no clusters*
Ex3	0.08±0.1	21.6±5.3	48.1±1.1	*no clusters*
Ex4	4.8±0.4	7.1±0.7	7.7±0.7	22.7± 5.9
Ex5	7.7±0.3	7.0±2.0	18.7±2.7	16.5± 3.9

$k = 2$; **Image**: 640 records, $N = 16$, $k = 15$; **Sonar**: 208 data points, $N = 60$, and $k = 2$.

To study whether our projected clustering algorithm is applicable to gene expression profiles, we used two data sets: the B-cell lymphoma [4] and the DNA microarray of gene expression profiles in hereditary breast cancer [10]. The lymphoma data set contains 96 samples, each with 4026 expression values. We clustered the samples with the expression values of the genes as attributes (4026 dimensions). The samples are categorized into 9 classes according to the category of mRNA sample studied. We used the class labels to compute error rates. Again, error rates are computed according to the confusion matrices.

We also experiment our algorithm with a DNA microarray of gene expression profiles in hereditary breast cancer [10]. The microarray contains expression levels of 3226 genes under 22 conditions. The data set is presented as a matrix: each row corresponds to a gene, and each column represents a condition under which the gene is developed. Biologists are interested in finding set of genes showing strikingly similar up-regulation and down-regulation under a set of conditions. We clustered the genes with the expression values of the samples as attributes (22 dimensions). Since class labels are not available for this data set, we utilize the mean squared residue score as defined in [6] to assess the quality of the clusters detected by LAC and PROCLUS algorithms. The lowest score value 0 indicates that the gene expression levels fluctuate in unison. The aim is to find biclusters with low mean squared residue score.

Results on Simulated Data. The performance results reported in Table 1 clearly demonstrate the large gain in performance obtained by the LAC algorithm with respect to PROCLUS and K-means with high dimensional data. The good performance of LAC on Examples 4 and 5 shows that our algorithm is able to detect clusters folded in subspaces not necessarily aligned with the input axes.

The large error rates of K-means for the 30 and 50 dimensional data sets (Examples 2 and 3) show how ineffective a distance function that equally use all input features can be in high dimensional spaces. PROCLUS requires the average number of dimensions per cluster as parameter in input; its value has to be at least two. We have cross-validated this parameter and report the best error rates obtained in Table 1. PROCLUS is able to select highly relevant features for low dimensional data, but fails to do so in higher dimensions, as the large error rates for Examples 2 and 3 show. The performance of PROCLUS is highly sensitive to the value of its input parameter. If the average number of dimensions is erroneously estimated, the performance of PROCLUS significantly worsens. This can be a serious problem with real data, when the required parameter value is most likely unknown.

We set the parameters of DOC as suggested in [11]. DOC failed to find any clusters in the high dimensional examples. In lower dimensions, DOC offered improvements over K-means, but it is considerably worst than LAC or PROCLUS. It is particularly hard to set the input parameters of DOC, as local variances of features are unknown in practice.

Results on Real Data. Table 2 reports the error rates obtained on the real data sets with class labels. For LAC we tested the integer values from 1 to 5 for the parameter h, and report the best error rates achieved. We ran PROCLUS with input parameter values from 2 to N for each data set, and report the best error rate obtained in each case. For the lymphoma data set (4026 dimensions) we tested several input parameter values of PROCLUS, and found the best result at 3500. LAC gives a better performance in each data set. In three cases (Breast, Pima, Image) LAC and K-means have very similar error rates. For these sets, LAC didn't find local structures in the data, and credited approximately equal weights to features. K-means performs poorly on the OQ and Sonar data. The enhanced performance given by the subspace clustering techniques in these two cases suggest that data are likely to be locally correlated. This seems to be true also for the lymphoma data. The DOC algorithm performed poorly. We did extensive testing for different parameter values, and report the best error rates in Table 2. DOC failed to find any clusters in the Lymphoma data set (4026 dimensions). These results clearly show the difficulty of using the DOC algorithm in practice.

We capture robustness of a technique by computing the ratio b_m of its error rate e_m and the smallest error rate over all methods being compared in a particular example: $b_m = e_m / \min_{1 \leq k \leq 3} e_k$. Figure 1 plots the distribution of b_m for each method over the six real data sets. For each method (LAC, PROCLUS, K-means) we stack the six b_m values. LAC is the most robust technique among the methods compared.

We run the LAC and PROCLUS algorithms using

Table 2: Average error rates for real data.

	LAC	*PROCLUS*	*K-means*	*DOC*
OQ	30.9	31.6	47.1	54.0
Breast	4.5	5.7	4.5	32.9
Pima	29.6	33.1	28.9	42.7
Image	39.1	42.5	38.3	45.8
Sonar	38.5	39.9	46.6	65.0
Lymphoma	32.3	33.3	39.6	–

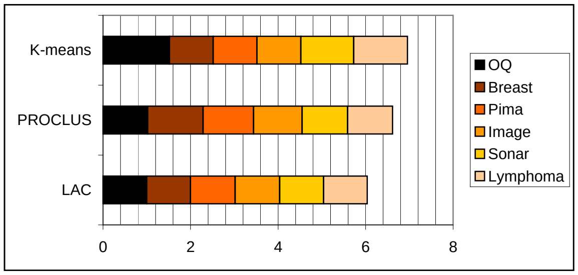

Figure 1: Performance distributions over real data sets

Table 3: Size, score and dimensions of the clusters detected by LAC and PROCLUS algorithms on the microarray data ($k = 3$ and $k = 4$).

$k = 3$	*LAC*	*PROCLUS*
C0 (size, score)	1220×5, **11.98**	1635×4, **9.41**
dimensions	9,13,14,19,22	7,8,9,13
C1 (size, score)	1052×5, **1.07**	1399×6, **48.18**
dimensions	7,8,9,13,18	7,8,9,13,19,22
C2 (size, score)	954×4, **5.32**	192×5, **2.33**
dimensions	12,13,16,18	2,7,10,19,22
$k = 4$	*LAC*	*PROCLUS*
C0 (size, score)	1701×5, **4.52**	1249×5, **3.90**
dimensions	7,8,9,19,22	7,8,9,13,22
C1 (size, score)	1255×5, **3.75**	1229×6, **42.74**
dimensions	7,8,9,13,22	7,8,9,13,19,22
C2 (size, score)	162 outliers	730×4, **15.94**
dimensions	-	7,8,9,13
C3 (size, score)	108 outliers	18×5, **3.97**
dimensions	-	6,11,14,16,21

the DNA microarray data and small values of k ($k = 3$ and $k = 4$). For this data set, DOC was not able to find any clusters. Table 3 shows sizes, scores, and dimensions of the biclusters detected by LAC and PROCLUS. For LAC we have selected the dimensions with the largest weights ($h = 9$). For $k = 3$, within each cluster four or five conditions received significant larger weight than the remaining ones. Hence, we selected those dimensions. By taking into consideration this result, we run PROCLUS with five as value of its input parameter. For $k = 4$, within two clusters five conditions receive again considerably larger weight than the others. The remaining two clusters contain fewer genes, and all conditions receive equal weights. Since no correlation was found among the conditions in these two cases, we have "labelled" the corresponding tuples as outliers.

Different combinations of conditions are selected for different biclusters, as also expected from a biological perspective. Some conditions are often selected, by both LAC and PROCLUS (e.g., conditions 7,8, and 9). The mean squared residue scores of the biclusters produced by LAC are consistently low, as desired. On the contrary, PROCLUS provides some clusters with higher scores (C1 in Table 3).

6 Conclusions

We have formalized the problem of finding different clusters in different subspaces. Our algorithm discovers clusters in subspaces spanned by different combinations of dimensions via local weightings of features. In our future work we will investigate the issue of noise in the data, to which the initial choice of centroids is sensitive. We will also study mechanisms to automatically setting the parameter h of our exponential weighting scheme.

References

[1] Aggarwal, C., Procopiuc, C., Wolf, J. L., Yu, P. S., and Park, J. S. Fast Algorithms for Projected Clustering. *SIGMOD*, 1999.

[2] Aggarwal, C. C., and Yu, P. S., Finding generalized projected clusters in high dimensional spaces. *SIGMOD*, 2000.

[3] Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *SIGMOD*, 1998.

[4] Alizadeh, A. A., et al. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

[5] Bottou, L., and Vapnik, V. Local learning algorithms. *Neural computation*, 4(6):888–900, 1992.

[6] Cheng, Y., and Church, G. M. Biclustering of expression data. *Int'l conference on intelligent systems for molecular biology*, 2000.

[7] Domeniconi, C. *Locally Adaptive Techniques for Pattern Classification*, PhD dissertation, UC Riverside, Computer Science Dept., August 2002.

[8] Duda, R. O., and Hart, P. E. *Pattern Classification and Scene Analysis. John Wiley and Sons*, 1973.

[9] Fukunaga, K. Introduction to Statistical Pattern Recognition. *Academic Press*, 1990.

[10] Hedenfalk, I., et al. Gene expression profiles in hereditary breast cancer, *N Engl J Med*, **344**:539–548, 2001.

[11] Procopiuc, C. M., Jones, M., Agarwal, P. K., and Murali, T. M. A Monte Carlo algorithm for fast projective clustering. *SIGMOD*, 2002.