Locally Adaptive Metrics for Clustering High Dimensional Data

Carlotta Domeniconi
George Mason University

Dimitrios Gunopulos
UC Riverside

Sheng Ma
IBM T.J. Watson Research Center

Dimitris Papadopoulos
UC Riverside

Bojun Yan
George Mason University

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 with respect to competitive methods, using both synthetic and real datasets. In particular, our results show the feasibility of the proposed technique to perform simultaneous clustering of genes and conditions in gene expression data, and clustering of very high dimensional data such as text data.

Keywords: subspace clustering, dimensionality reduction, local feature relevance, gene expression data, text data.

1. Introduction

The clustering problem concerns the discovery of homogeneous groups of data according to a certain similarity measure. It has been studied extensively in statistics (Arabie and Hubert, 1996), machine learning (Cheeseman and Stutz, 1996; Michalski and Stepp, 1996), and database communities (Ng and Han, 1994; Ester et al., 1995; Zhang et al., 1996).

Given a set of multivariate data, (partitional) 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. The popular K-means or K-medoids methods compute one representative point per cluster, and assign each object to the cluster with the closest

representative, so that the sum of the squared differences between the objects and their representatives is minimized. Finding a set of representative vectors for clouds of multi-dimensional data is an important issue in data compression, signal coding, pattern classification, and function approximation tasks.

Clustering suffers from the curse of dimensionality problem in high dimensional spaces. 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.

The problem of high dimensionality could be addressed by requiring the user to specify a subspace (i.e., subset of dimensions) for cluster analysis. However, the identification of subspaces by the user is an error-prone process. More importantly, correlations that identify clusters in the data are likely not to be known by the user. Indeed, we desire such correlations, and induced subspaces, to be part of the findings of the clustering process itself.

An alternative 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) (or Karhunen-Loeve transformation) (Duda and Hart, 1973; Fukunaga, 1990) 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 input space. Local feature selection allows to embed different distance measures 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 along which data are strongly correlated receive a large weight, that has the effect of constricting distances along that dimension. Figure 1 gives a simple example. The left plot depicts two clusters of data elongated along the x and y dimensions. The right plot shows the same clusters, where within-cluster distances between points are computed using the respective local weights generated by our algorithm. The weight values reflect local correlations of data, and reshape each cluster as a *dense spherical cloud*. This directional local reshaping of distances better separates clusters, and allows for the discovery of different patterns in different subspaces of the original input space.

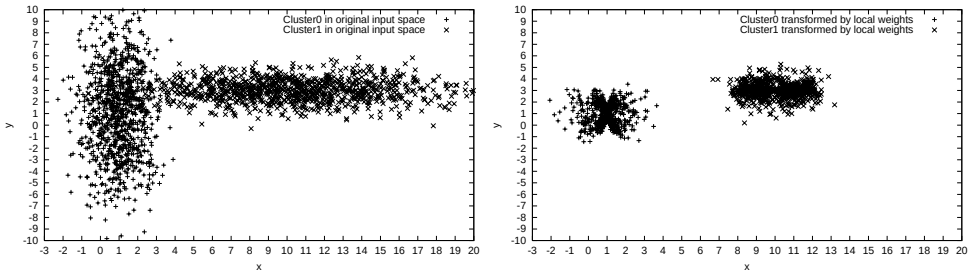

Figure 1. (Left) Clusters in original input space. (Right) Clusters transformed by local weights.

1.1. Our Contribution

An earlier version of this work appeared in (Domeniconi et al., 2004). However, this paper is a substantial extension, which includes (as new material) a new derivation and motivation of the proposed algorithm, a proof of convergence of our approach, a variety of experiments, comparisons, and analysis using high-dimensional text and gene expression data. Specifically, the contributions of this paper are as follows:

1. We formalize the problem of finding different clusters in different subspaces. Our algorithm (Locally Adaptive Clustering, or LAC) 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.

2. The output of our algorithm is twofold. It provides a partition of the data, so that the points in each set of the partition constitute a cluster. In addition, each set is associated with a weight vector, whose values give information of the degree of relevance of features for each partition.

3. We formally prove that our algorithm converges to a local minimum of the associated error function, and experimentally demonstrate the gain in perfomance we achieve with our method. In particular, our results show the feasibility of the proposed technique to perform simultaneous clustering of genes and conditions in gene expression data, and clustering of very high dimensional data such as text data.

2. Related Work

Local dimensionality reduction approaches for the purpose of efficiently indexing high dimensional spaces have been recently discussed in the database literature (Keogh et al., 2001; Chakrabarti and Mehrotra, 2000; Thomasian et al., 1998). Applying global dimensionality reduction techniques when data are not globally correlated can cause significant loss of distance information, resulting in a large number of false positives and hence a high query cost. The general approach adopted by the authors is to find local correlations in the data, and perform dimensionality reduction on the locally correlated clusters individually. For example, in (Chakrabarti and Mehrotra, 2000), the authors first construct spacial clusters in the original input space using a simple tecnique that resembles K-means. Principal component analysis is then performed on each spatial cluster individually to obtain the principal components.

In general, the efficacy of these methods depends on how the clustering problem is addressed in the first place in the original feature space. A potential serious problem with such techniques is the lack of data to locally perform PCA on each cluster to derive the principal components. Moreover, for clustering purposes, the new dimensions may be difficult to interpret, making it hard to understand clusters in relation to the original space.

The problem of finding different clusters in different subspaces of the original input space has been addressed in (Agrawal et al., 1998). 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. To approximate the density of the points, the input space is partitioned into cells by dividing each dimension into the same number ξ of equal length intervals. For a given set of dimensions, the cross product of the corresponding intervals (one for each dimension in the set) is called a *unit* in the respective subspace. A unit is dense if the number of points it contains is above a given threshold τ. Both ξ and τ are parameters defined by the user. The algorithm finds all dense units in each k-dimensional subspace by building from the dense units of $(k-1)$-dimensional subspaces, and then connects them to describe the clusters as union of maximal rectangles.

While the work in (Agrawal et al., 1998) successfully introduces a methodology for looking at different subspaces for different clusters, it does not compute a partitioning of the data into disjoint groups. The reported dense regions largely overlap, since for a given dense region all its projections on lower dimensionality subspaces are also dense, and they all get reported. 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 (Procopiuc et al., 2002), 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.

(Dy and Brodley, 2000) also addresses the problem of feature selection to find clusters hidden in high dimensional data. The authors search through feature subset spaces, evaluating each subset by first clustering in the corresponding subspace, and then evaluating the resulting clusters and feature subset using the chosen feature selection criterion. The two feature selection criteria investigated are the scatter separability used in discriminant analysis (Fukunaga, 1990), and a maximum likelihood criterion. A sequential forward greedy strategy (Fukunaga, 1990) is employed to search through possible feature subsets. We observe that dimensionality reduction is performed globally in this case. Therefore, the technique in (Dy and Brodley, 2000) is expected to be effective when a dataset contains some relevant features and some irrelevant (noisy) ones, across all clusters.

The problem of finding different clusters in different subspaces is also addressed in (Aggarwal et al., 1999). 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. PROCLUS starts with choosing a random set of medoids, and then progressively improves the quality of medoids by performing an iterative hill climbing procedure that discards the 'bad' medoids from the current set. In order to find the set of dimensions that matter the most for each cluster, the algorithm selects the dimensions along which the points have the smallest average distance from the current medoid. ORCLUS (Aggarwal and Yu, 2000) modifies the PROCLUS algorithm by adding a merging process of clusters, and selecting for each cluster principal components instead of attributes.

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. For example, if data of two clusters in two dimensions are distributed as in Figure 1 (Left), PROCLUS may find that feature x is the most important for cluster 0, and feature y is the most important for cluster 1. But projecting cluster 1 along the y dimension doesn't allow to properly separate points of the two clusters. We avoid this problem by keeping both dimensions for both clusters, and properly weighting distances along each feature within each cluster.

The problem of feature weighting in K-means clustering has been addressed in (Modha and Spangler, 2003). Each data point is represented as a collection of vectors, with "homogeneous" features within each measurement space. The objective is to determine one (global) weight value for each feature space. The optimality criterion pursued is the minimization of the (Fisher) ratio between the average within-cluster distortion and the average between-cluster distortion. However, the proposed method does *not learn* optimal weights from the data. Instead, different weight value combinations are ran through a K-means-like algorithm, and the combination that results in the lowest Fisher ratio is chosen. We also observe that the weights as defined in (Modha and Spangler, 2003) are global, in constrast to ours which are local to each cluster.

Recently (Dhillon et al., 2003), a theoretical formulation of subspace clustering based on information theory has been introduced. The data contingency matrix (e.g., document-word co-occurrence matrix) is seen as an empirical joint probability distribution of two discrete ran-

dom variables. Subspace clustering is then formulated as a constrained optimization problem where the objective is to maximize the mutual information between the clustered random variables.

Generative approaches have also been developed for local dimensionality reduction and clustering. The approach in (Ghahramani and Hinton, 1996) makes use of maximum likelihood factor analysis to model local correlations between features. The resulting generative model obeys the distribution of a mixture of factor analyzers. An expectation-maximization algorithm is presented for fitting the parameters of the mixture of factor analyzers. The choice of the number of factor analyzers, and the number of factors in each analyzer (that drives the dimensionality reduction) remain an important open issue for the approach in (Ghahramani and Hinton, 1996).

(Tipping and Bishop, 1999) extends the single PCA model to a mixture of local linear sub-models to capture nonlinear structure in the data. A mixture of principal component analyzers model is derived as a solution to a maximum-likelihood problem. An EM algorithm is formulated to estimate the parameters.

While the methods in (Ghahramani and Hinton, 1996; Tipping and Bishop, 1999), as well as the standard mixture of Gaussians technique, are generative and parametric, our approach can be seen as an attempt to directly estimate from the data local correlations between features. Furthermore, both mixture models in (Ghahramani and Hinton, 1996; Tipping and Bishop, 1999) inherit the soft clustering component of the EM update equations. On the contrary, LAC computes a partitioning of the data into disjoint groups. As previously mentioned, for many data mining applications a partition of the data is desirable since it provides a clear interpretability of the results. We finally observe that, while mixture of Gaussians models, with arbitrary covariance matrices, could in principle capture local correlations along any directions, lack of data to locally estimate full covariance matrices in high dimensional spaces is a serious problem in practice.

2.1. Biclustering of Gene Expression Data

Microarray technology is one of the latest breakthroughs in experimental molecular biology. Gene expression data are generated by DNA chips and other microarray techniques, and they are often presented as matrices of expression levels of genes under different conditions (e.g., environment, individuals, tissues). 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.

To this extent, recently the concept of *bicluster* has been introduced (Cheng and Church, 2000). A bicluster is a subset of genes and a subset of conditions with a high similarity score. A particular score that applies to expression data is the *mean squared residue score* (Cheng and Church, 2000). Let I and J be subsets of genes and experiments respectively. The pair (I, J) specifies a submatrix A_{IJ} with a mean squared residue score defined as follows:

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2, \tag{1}$$

where $a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}$, $a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij}$, and $a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij}$. They represent the row and column means, and the mean of the submatrix, respectively. The lowest score $H(I, J) = 0$ indicates that the gene expression levels fluctuate in unison. The aim is then to find biclusters with low mean squared residue score (below a certain threshold).

We observe that the mean squared residue score is minimized when subsets of genes and experiments (or dimensions) are chosen so that the gene vectors (i.e., rows of the resulting bicluster) are close to each other with respect to the Euclidean distance. As a result, the LAC algorithm, and other subspace clustering algorithms, are well suited to perform simultaneous clustering of both genes and conditions in a microarray data matrix. (Wang et al., 2002) introduces an algorithm (pCluster) for clustering similar patterns, that has been applied to DNA microarray data of a type of yeast. The pCluster model optimizes a criterion that is different from the mean squared residue score, as it looks for coherent patterns on a subset of dimensions (e.g., in an identified subspace, objects reveal larger values for the second dimension than for the first).

3. Locally Adaptive Metrics for Clustering

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 dataset.

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* or *centers*. 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 $\{S_1, \ldots, S_k\}$:

$$S_j = \{\mathbf{x} | (\sum_{i=1}^{N} w_{ji}(x_i - c_{ji})^2)^{1/2} < (\sum_{i=1}^{N} w_{li}(x_i - c_{li})^2)^{1/2}, \forall l \neq j\}, \quad (2)$$

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:

$$E_1(C, W) = \sum_{j=1}^{k} \sum_{i=1}^{N} (w_{ji} \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2) \quad (3)$$

subject to the constraints $\sum_i w_{ji} = 1 \; \forall j$. 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 $W = [\mathbf{w}_1 \ldots \mathbf{w}_k]$. For shortness of notation, we set

$$X_{ji} = \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2,$$

where $|S_j|$ is the cardinality of set S_j. X_{ji} represents the average distance from the centroid $\mathbf{c}_j$ of points in cluster j along dimension i. The solution

$$(C^*, W^*) = argmin_{(C,W)} E_1(C, W)$$

will discover one dimensional clusters: it will put maximal (i.e., unit) weight on the feature with smallest dispersion X_{ji} within each cluster j, and zero weight on all other features. Our objective, instead, is to find weighted multidimensional clusters, where the unit weight gets distributed among all features according to the respective dispersion of data within each cluster. One way to achieve this goal is to add the regularization term $\sum_{i=1}^{N} w_{ji} log w_{ji}$ [1], which represents the negative entropy of the weight distribution for each cluster (Friedman and

[1] Different regularization terms lead to different weighting schemes.

Meulman, 2002). It penalizes solutions with maximal (unit) weight on the single feature with smallest dispersion within each cluster. The resulting error function is

$$E_2(C,W) = \sum_{j=1}^{k}\sum_{i=1}^{N}(w_{ji}X_{ji} + hw_{ji}logw_{ji}) \tag{4}$$

subject to the same constraints $\sum_i w_{ji} = 1\ \forall j$. The coefficient $h \geq 0$ is a parameter of the procedure; it controls the relative differences between feature weights. In other words, h controls how much the distribution of weight values will deviate from the uniform distribution. We can solve this constrained optimization problem by introducing the Lagrange multipliers λ_j (one for each constraint), and minimizing the resulting (unconstrained now) error function

$$E(C,W) = \sum_{j=1}^{k}\sum_{i=1}^{N}(w_{ji}X_{ji} + hw_{ji}logw_{ji}) + \sum_{j=1}^{k}\lambda_j(1 - \sum_{i=1}^{N}w_{ji}) \tag{5}$$

For a fixed partition P and fixed c_{ji}, we compute the optimal w^*_{ji} by setting $\frac{\partial E}{\partial w_{ji}} = 0$ and $\frac{\partial E}{\partial \lambda_j} = 0$. We obtain:

$$\frac{\partial E}{\partial w_{ji}} = X_{ji} + hlogw_{ji} + h - \lambda_j = 0 \tag{6}$$

$$\frac{\partial E}{\partial \lambda_j} = 1 - \sum_{i=1}^{N} w_{ji} = 0 \tag{7}$$

Solving equation (6) with respect to w_{ji} we obtain $hlogw_{ji} = -X_{ji} + \lambda_j - h$. Thus:

$$\begin{aligned} w_{ji} = exp(-X_{ji}/h + (\lambda_j/h) - 1) &= exp(-X_{ji}/h)exp((\lambda_j/h) - 1) \\ &= \frac{exp(-X_{ji}/h)}{exp(1 - \lambda_j/h)}. \end{aligned}$$

Substituting this expression in equation (7):

$$\frac{\partial E}{\partial \lambda_j} = 1 - \sum_{i=1}^{N}\frac{exp(-X_{ji}/h)}{exp(1-\lambda_j/h)} = 1 - \frac{1}{exp(-\lambda_j/h)}\sum_{i=1}^{N}exp((-X_{ji}/h) - 1) = 0.$$

Solving with respect to λ_j we obtain

$$\lambda_j = -hlog\sum_{i=1}^{N}exp((-X_{ji}/h) - 1).$$

Thus, the optimal w_{ji}^* is

$$\begin{aligned} w_{ji}^* &= \frac{exp(-X_{ji}/h)}{exp(1 + log(\sum_{i=1}^{N} exp((-X_{ji}/h) - 1)))} \\ &= \frac{exp(-X_{ji}/h)}{\sum_{i=1}^{N} exp(-X_{ji}/h)} \end{aligned} \tag{8}$$

For a fixed partition P and fixed w_{ji}, we compute the optimal c_{ji}^* by setting $\frac{\partial E}{\partial c_{ji}} = 0$. We obtain:

$$\frac{\partial E}{\partial c_{ji}} = w_{ji} \frac{1}{|S_j|} 2 \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i) = \frac{2w_{ji}}{|S_j|} (|S_j| c_{ji} - \sum_{\mathbf{x} \in S_j} x_i) = 0.$$

Solving with respect to c_{ji} gives

$$c_{ji}^* = \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} x_i. \tag{9}$$

Solution (8) puts increased weight on features along which the dispersion X_{ji} is smaller, within each cluster. The degree of this increase is controlled by the value h. Setting $h = 0$, places all weight on the feature i with smallest X_{ji}, whereas setting $h = \infty$ forces all features to be given equal weight for each cluster j. By setting $E_0(C) = \frac{1}{N} \sum_{j=1}^{k} \sum_{i=1}^{N} X_{ji}$, we can formulate this result as follows.
Proposition: When $h = 0$, the error function E_2 (4) reduces to E_1 (3); when $h = \infty$, the error function E_2 reduces to E_0.

4. Locally Adaptive Clustering Algorithm

We need to provide a search strategy to find a partition P that identifies the solution clusters. Our approach progressively improves the quality of initial centroids and weights, by investigating the space near the centers to estimate the dimensions that matter the most. Specifically, we proceed as follows.

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 $1/N$. 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 X_{ji} along each dimension from the points in S_j to $\mathbf{c}_j$. 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), as given in equation (8). The exponential weighting is more sensitive to changes in local feature relevance (Bottou and Vapnik, 1992) 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 as given in equation (9). The procedure is iterated until convergence is reached. 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 $w_{ji} = 1/N$, 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 $S_j = \{\mathbf{x} | j = arg\min_l L_w(\mathbf{c}_l, \mathbf{x})\}$, where $L_w(\mathbf{c}_l, \mathbf{x}) = (\sum_{i=1}^{N} w_{li}(c_{li} - x_i)^2)^{1/2}$;
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 $w_{ji} = \frac{exp(-X_{ji}/h)}{\sum_{l=1}^{N} exp(-X_{jl}/h)}$;
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.

The sequential structure of the LAC algorithm is analogous to the mathematics of the *EM* algorithm (Dempster et al., 1977; Wu, 1983). The hidden variables are the assignments of the points to the centroids. Step 3 constitutes the *E* step: it finds the values of the hidden variables S_j given the previous values of the parameters w_{ji} and c_{ji}. The following step (*M* step) consists in finding new matrices of weights and centroids that minimize the error function with respect to the current estimation

of hidden variables. It can be shown that the LAC algorithm converges to a local minimum of the error function (5). The running time of one iteration is $O(kDN)$.

We point out that the LAC algorithm can identify a degenerate solution, i.e. a partition with empty clusters, during any iteration. Although we didn't encounter this problem in our experiments, strategies developed in the literature, such as the insertion strategy (Mladenović and Brimberg, 1996), can be easily incorporated in our algorithm. In particular, we can proceed as follows: if the number of non-empty clusters in the current iteration of LAC is $l < k$, we can identify the l points with the leargest (weighted) distance to their cluster's centroid, and form l new clusters with a single point in each of them. The resulting non-degenerate solution is clearly better than the degenerate one since the selected l points give the largest contributions to the cost function, but it could possibly be improved. Therefore, the LAC iterations can continue until convergence to a non-degenerate solution.

5. Convergence of the LAC Algorithm

In light of the remark made above on the analogy of LAC with *EM* (Wu, 1983), here we prove that our algorithm converges to a solution that is a local minimum of the error function (5). To obtain this result we need to show that the error function decreases at each iteration of the algorithm. By derivation of equations (8) and (9), Step 4 and Step 6 of the LAC algorithm perform a gradient descent over the surface of the error function (5). We make use of this observation to show that each iteration of the algorithm decreases the error function. We prove the following theorem.

Theorem. The LAC algorithm converges to a local minimum of the error function (5).

Proof. For a fixed partition P and fixed c_{ji}, the optimal w'_{ji} obtained by setting $\frac{\partial E}{\partial w_{ji}} = 0$ and $\frac{\partial E}{\partial \lambda_j} = 0$ is:

$$w'_{ji} = \frac{exp(-X_{ji}/h)}{\sum_{l=1}^{N} exp(-X_{jl}/h)} \tag{10}$$

as in Step 4 of the LAC algorithm.

For a fixed partition P and fixed w_{ji}, the optimal c'_{ji} obtained by setting $\frac{\partial E}{\partial c_{ji}} = 0$ is:

$$c'_{ji} = \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} x_i \tag{11}$$

as in Step 6 of the LAC algorithm.

The algorithm consists in repeatedly replacing w_{ji} and c_{ji} with w'_{ji} and c'_{ji} using equations (10) and (11), respectively. The value of the error function E at completion of each iteration is $E_{P_1}(C', W')$, where we explicit the dependence of E on the partition of points P_1 computed in Step 5 of the algorithm. C' and W' are the matrices of the newly computed centroids and weights. Since the new partition P' computed in Step 3 of the successive iteration is by definition the best assignment of points $\mathbf{x}$ to the centroids c'_{ji} according to the weighted Euclidean distance with weights w'_{ji}, we have the following inequality:

$$E_{P'}(C', W') - E_{P_1}(C', W') \leq 0 \tag{12}$$

Using this result, and the identities $E(C', W') = E_{P'}(C', W')$ and $E(C, W) = E_P(C, W)$, we can derive the following inequality:

$$\begin{aligned} E(C', W') - E(C, W) &= E_{P'}(C', W') - E_{P_1}(C', W') \\ &+ E_{P_1}(C', W') - E_P(C, W) \\ &\leq E_{P_1}(C', W') - E_P(C, W) \leq 0 \end{aligned}$$

where the last inequality is derived by using the definitions of w'_{ji} and c'_{ji}.

Thus, each iteration of the algorithm decreases the lower bounded error function E (5) until the error reaches a fixed point where conditions $\mathbf{w}_j^{*\prime} = \mathbf{w}_j^*$, $\mathbf{c}_j^{*\prime} = \mathbf{c}_j^*$ $\forall j$ are verified. The fixed points $\mathbf{w}_j^*$ and $\mathbf{c}_j^*$ give a local minimum of the error function E.

6. Experimental Evaluation

In our experiments we have designed five different simulated datasets to compare the competitive algorithms under different conditions. Clusters are distributed according to multivariate gaussians with different mean and standard deviation vectors. We have tested problems with two and three clusters up to 50 dimensions. For each problem, we have generated five or ten training datasets, and for each of them an independent test set. In the following we report accuracy and performance results obtained via 5(10)-fold cross-validation comparing LAC, PROCLUS, DOC, K-means, and EM (Dempster et al., 1977) (mixture of Gaussians with diagonal - EM(d) - and full - EM(f) - covariance matrices). Among the subspace clustering techniques available in the literature, we chose PROCLUS (Aggarwal et al., 1999) and DOC (Procopiuc et al., 2002) since, as the LAC algorithm, they also compute a partition of the data. On the contrary, the CLIQUE technique (Agrawal

et al., 1998) allows overlapping between clusters, and thus its results are not directly comparable with ours.

Error rates are computed according to the confusion matrices that are also reported. For LAC we tested the integer values from 1 to 11 for the parameter $1/h$, and report the best error rates achieved. The k centroids are initialized by choosing well-scattered points among the given data. The mean vectors and covariance matrices provided by K-means are used to initialize the parameters of EM.

6.1. Simulated Data

1. **Example1**. The dataset consists of $N = 2$ input features and $k = 3$ clusters. All three clusters are distributed according to multivariate gaussians. Mean vector and standard deviations for one cluster are $(2, 0)$ and $(4, 1)$ respectively. For the second cluster the vectors are $(10, 0)$ and $(1, 4)$, and for the third are $(18, 0)$ and $(4, 1)$. Table I shows the results for this problem. We generated 60000 data points, and performed 10-fold cross-validation with 30000 training data and 30000 testing data.

2. **Example2**. This dataset consists of $N = 30$ input features and $k = 2$ clusters. Both clusters are distributed according to multivariate gaussians. Mean vector and standard deviations for one cluster are $(1, \ldots, 1)$ and $(10, 5, 10, 5, \ldots, 10, 5)$, respectively. For the other cluster the vectors are $(2, 1, \ldots, 1)$ and $(5, 10, 5, 10, \ldots, 5, 10)$. Table I shows the results for this problem. We generated 10000 data points, and performed 10-fold cross-validation with 5000 training and 5000 testing data.

3. **Example3**. This dataset consists of $N = 50$ input features and $k = 2$ clusters. Both clusters are distributed according to multivariate gaussians. Mean vector and standard deviations for one cluster are $(1, \ldots, 1)$ and $(20, 10, 20, 10, \ldots, 20, 10)$, respectively. For the other cluster the vectors are $(2, 1, \ldots, 1)$ and $(10, 20, 10, 20, \ldots, 10, 20)$. Table I shows the results for this problem. We generated 10000 data points, and performed 10-fold cross-validation with 5000 training data and 5000 testing data.

4. **Example4**. This dataset consists of off-axis oriented clusters, with $N = 2$ and $k = 2$. Figure 3 shows the distribution of the points for this dataset. We generated 20000 data points, and performed 5-fold-cross-validation with 10000 training data and 10000 testing data. Table I shows the results.

5. **Example5**. This dataset consists again of off-axis oriented two dimensional clusters. This dataset contains three clusters, as Figure 4

depicts. We generated 30000 data points, and performed 5-fold-cross-validation with 15000 training data and 15000 testing data. Table I shows the results.

6.2. Real Data

We used ten real datasets. The OQ-letter, Wisconsin breast cancer, Pima Indians Diabete, and Sonar data are taken from the UCI Machine Learning Repository. The Image data set is obtained from the MIT Media Lab. We used three high dimensional text datasets: Classic3, Spam2000, and Spam5996. The documents in each dataset were preprocessed by eliminating stop words (based on a stop words list), and stemming words to their root source. We use as feature values for the vector space model the frequency of the terms in the corresponding document. The Classic3 dataset is a collection of abstracts from three categories: MEDLINE (abstracts from medical journals), CISI (abstracts from IR papers), CRANFIELD (abstracts from aerodynamics papers). The Spam data belong to the Email-1431 dataset. This dataset consists of emails falling into three categories: conference (370), jobs (272), and spam (786). We run two different experiments with this dataset. In one case we reduce the dimensionality to 2000 terms (Spam2000), in the second case to 5996 (Spam5996). In both cases we consider two clusters by merging the conference and jobs mails into one group (non-spam). The characteristics of these eight datasets are as follows. *OQ*: $D = 1536$, $N = 16$, $k = 2$; *Breast*: $D = 683$, $N = 9$, $k = 2$; *Pima*: $D = 768$, $N = 8$, $k = 2$; *Image*: $D = 640$, $N = 16$, $k = 15$; *Sonar*: $D = 208$, $N = 60$, $k = 2$; *Classic3*: $D = 3893$, $N = 3302$, $k = 3$; *Spam2000*: $D = 1428$, $N = 2000$, $k = 2$; *Spam5996*: $D = 1428$, $N = 5996$, $k = 2$. To study whether our projected clustering algorithm is applicable to gene expression profiles, we used two datasets: the B-cell lymphoma (Alizadeh et al., 2000) and the DNA microarray of gene expression profiles in hereditary breast cancer (Hedenfalk et al., 2001). The lymphoma dataset 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, according to the confusion matrices. We also experiment our algorithm with a DNA microarray dataset (Hedenfalk et al., 2001). The microarray contains expression levels of 3226 genes under 22 conditions. We clustered the genes with the expression values of the samples as attributes (22 dimensions). The dataset is presented as a matrix: each row corresponds to a gene, and each column represents a condition under which the gene is developed.

Table I. Average error rates for simulated data.

	LAC	*PROCLUS*	*K-means*	*DOC*	*EM (d)*	*EM (f)*
Ex1	11.4±0.3	13.8±0.7	24.2±0.5	35.2± 2.2	**5.1** ± 0.4	**5.1** ± 0.4
Ex2	**0.5**±0.4	27.9±9.8	48.4±1.1	*no clusters*	0.6 ±0.3	0.8 ±0.3
Ex3	0.08±0.1	21.6±5.3	48.1±1.1	*no clusters*	**0.0** ±0.1	25.5 ±0.2
Ex4	4.8±0.4	7.1±0.7	7.7±0.7	22.7± 5.9	4.8 ± 0.2	**2.3** ± 0.2
Ex5	7.7±0.3	7.0±2.0	18.7±2.7	16.5± 3.9	6.0 ± 0.2	**2.3** ± 0.2
Average	4.9	15.5	29.4	24.8	**3.3**	7.2

Biologists are interested in finding set of genes showing strikingly similar up-regulation and down-regulation under a set of conditions. Since class labels are not available for this dataset, we utilize the mean squared residue score as defined in (1) 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 (in general, below a certain threshold).

Table II. Average number of iterations.

	LAC	*PROCLUS*	*K-means*	*EM(d)*
Ex1	7.2	6.7	16.8	22.4
Ex2	3.2	2.0	16.1	6.3
Ex3	3.0	4.4	19.4	6.0
Ex4	7.0	6.4	8.0	5.6
Ex5	7.8	9.8	15.2	27.6
Average	**5.6**	5.9	15.1	13.6

6.3. Results on Simulated Data

The performance results reported in Table I 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. Figures 5 and 6 illustrate the results obtained with

Table III. Dimensions selected by PROCLUS.

	C0	*C1*	*C2*
Ex1	2,1	2,1	2,1
Ex2	8,30	19,15,21,1,27,23	-
Ex3	50,16	50,16,17,18,21,22,23,19,11,3	-
Ex4	1,2	2,1	-
Ex5	1,2	2,1	1,2

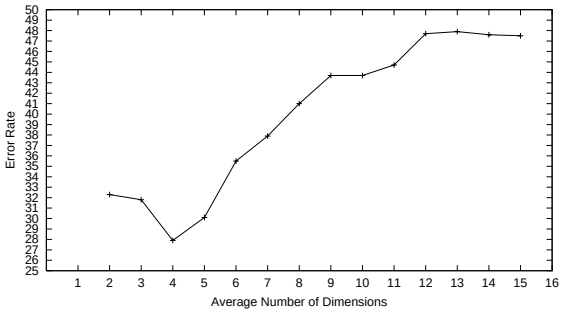

Figure 2. Example2: Error rate of PROCLUS versus Average number of dimensions.

Table IV. Confusion matrices for Example1.

LAC	*C0* (input)	*C1* (input)	*C2* (input)
C0 (output)	8315	0	15
C1 (output)	1676	10000	1712
C2 (output)	9	0	8273
PROCLUS	*C0* (input)	*C1* (input)	*C2* (input)
C0 (output)	7938	0	7
C1 (output)	2057	10000	2066
C2 (output)	5	0	7927
K-means	*C0* (input)	*C1* (input)	*C2* (input)
C0 (output)	9440	4686	400
C1 (output)	411	3953	266
C2 (output)	149	1361	9334

Table V. LAC: Weight values for Example1.

Cluster	*Std1*	*Std2*	w_1	w_2
C0	4	1	0.46	0.54
C1	1	4	0.99	0.01
C2	4	1	0.45	0.55

LAC and K-means, respectively, on Example 5. The large error rates of K-means for the 30 and 50 dimensional datasets (Examples 2 and 3) show how ineffective a distance function that equally use all input features can be in high dimensional spaces. Example 1, 2, and 3 offer optimal conditions for EM(d); Example 4 and 5 are optimal for EM(f). As a consequence, EM(d) and EM(f) provide best error rates in such respective cases. Nevertheless, LAC gives error rates similar to EM(d) under conditions which are optimal for the latter, especially in higher dimensions. The large error rate of EM(f) for Example 3 shows the difficulty of estimating full covariance matrices in higher dimensions.

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 I. PROCLUS is able to select highly relevant features for datasets in low dimensions, but fails to do so in higher dimensions, as the large error rates for Examples 2 and 3 show. Table III shows the dimensions selected by PROCLUS for each dataset and each cluster. Figure 2 plots the error rate as a function of the average number of dimensions per cluster, obtained by running PROCLUS on Example 2. The best error rate (27.9%) is achieved in correspondence of the value four. The error rate worsens for larger values of the average number of dimensions. Figure 2 shows that 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 (Procopiuc et al., 2002). DOC failed to find any clusters in the high dimensional examples. It is particularly hard to set the input parameters of DOC, as local variances of features are unknown in practice.

Table II shows the average number of iterations performed by LAC, K-means, and EM(d) to achieve convergence, and by PROCLUS to achieve the termination criterion. For each problem, the rate of con-

vergence of LAC is superior to the rate of K-means: on Examples 1 through 5 the speed-ups are 2.3, 5.0, 6.5, 1.1, and 1.9 respectively. The number of iterations performed by LAC and PROCLUS is close for each problem, and the running time of an iteration of both algorithms is $O(kDN)$ (where k is the number of clusters, D is the number of data points, and N the number of dimensions). The faster rate of convergence achieved by the LAC algorithm with respect to K-means (and EM(d)) is motivated by the exponential weighting scheme provided by equation (8), which gives the optimal weight values w_{ji}^*. Variations of the within-cluster dispersions X_{ji} (along each dimension i) are exponentially reflected into the corresponding weight values w_{ji}. Thus the (exponential) weights are more sensitive (than quadratic or linear ones, for example) to changes in local feature relevance. As a consequence, a minimum value of the error function (5) can be reached in less iterations than the corresponding unweighted cost function minimized by the K-means algorithm.

To further test the accuracy of the algorithms, for each problem we have computed the *confusion matrices*. The entry (i, j) in each confusion matrix is equal to the number of points assigned to output cluster i, that were generated as part of input cluster j. We also report the average weight values per cluster obtained over the runs conducted in our experiments. Results are reported in Tables IV-XI.

Tables V, IX, and XI show that there is a perfect correspondence between the weight values of each cluster and the correlation patterns of data within the same cluster. This is of great importance for applications that require not only a good partitioning of data, but also information to what features are relevant for each partition.

As expected, the resulting weight values for one cluster depends on the configurations of other clusters as well. If clusters have the same standard deviation along one dimension i, they receive almost identical weights for measuring distances along that feature. This is informative of the fact that feature i is equally relevant for both partitions. On the other hand, weight values are largely differentiated when two clusters have different standard deviation values along the same dimension i, implying different degree of relevance of feature i for the two partitions (see for example Table V).

6.4. Results on Real Data

Table XII reports the error rates obtained on the nine real datasets with class labels. For LAC we fixed the value of the parameter $1/h$ to 9 (this value gave in general good results with the simulated data). We ran PROCLUS with input parameter values from 2 to N for each

dataset, and report the best error rate obtained in each case. For the Lymphoma (4026 dimensions), Classic3 (3302 dimensions), Spam2000 (2000 dimensions), and Spam5996 (5996 dimensions) we tested several input parameter values of PROCLUS, and found the best result at 3500, 350, 170, and 300 respectively. LAC gives the best error rate in seven out of the nine datasets. LAC outperforms PROCLUS and EM(d) in each dataset. EM does not perform well in general, and particularly in higher dimensions. This is likely due to the non-Gaussian distributions of real data. EM(f) (Netlab library for Matlab) failed to run to completion on the very high dimensional data due to memory problems. In three cases (Breast, Pima, Image), LAC and K-means have very similar error rates. For these sets, LAC did not 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 LAC algorithm did extremely well on the three high dimensional text data (Classic3, Spam2000, and Spam5996), which demostrate the capability of LAC in finding meaningful local structure in high dimensional spaces. This result suggests that an analysis of the weights credited to terms can guide the automatic identification of class-specific keywords, and thus the process of label assignment to clusters.

The DOC algorithm performed poorly, and failed to find any clusters on the very high dimensional data (Lymphoma, Classic3, Spam2000, and Spam5996). We did extensive testing for different parameter values, and report the best error rates in Table XII. For the OQ data, we tested width values from 0.1 to 3.4 (at steps of 0.1). (The two actual clusters in this dataset have standard deviation values along input features in the ranges $(0.7, 3.2)$ and $(0.95, 3.2)$.) The best result obtained reported one cluster only, and 63.0% error rate. We also tried a larger width value (6), and obtained one cluster again, and error rate 54.0%. For the Sonar data we obtained the best result reporting two clusters for a width value of 0.5. Though, the error rate is still very high (65%). We tested several other values (larger and smaller), but they all failed to finding any cluster in the data. (The two actual clusters in this dataset have standard deviation values along input features in the ranges $(0.005, 0.266)$ and $(0.0036, 0.28)$.) 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: $b_m = e_m / \min_{1 \leq k \leq 4} e_k$. Thus, the best method m^* for an example has $b_{m^*} = 1$, and all other methods

have larger values $b_m \geq 1$, for $m \neq m^*$. The larger the value of b_m, the worse the performance of method m is in relation to the best one for that example, among the methods being compared. The distribution of the b_m values for each method m over all the examples, therefore, seems to be a good indicator concerning its robustness. For example, if a particular method has an error rate close to the best in every problem, its b_m values should be densely distributed around the value 1. Any method whose b value distribution deviates from this ideal distribution reflect its lack of robustness. Figure 10 plots the distribution of b_m for each method over the six real datasets OQ, Breast, Pima, Image, Sonar and Lymphoma. For scaling issues, we plot the distribution of b_m for each method over the three text data separately in Figure 11. For each method (LAC, PROCLUS, K-means, EM(d)) we stack the six b_m values. (We did not consider DOC since it failed to find reasonable patterns in most cases.) LAC is the most robust technique among the methods compared. In particular, Figure 11 graphically depicts the strikingly superior performance of LAC over the text data with respect to the competitive techniques.

To investigate the false positive and false negative rates on the spam data we show the corresponding confusion matrices in Tables XIII and XIV. In both cases, LAC has low false positive (FP) and low false negative (FN) rates. On Spam2000: $FP = 0.26\%$, $FN = 2.3\%$ On Spam5996: $FP = 2.66\%$, $FN = 7.85\%$. PROCLUS discovers, to some extent, the structure of the two groups for Spam2000 ($FP = 18.8\%$, $FN = 35.1\%$), but fails completely for Spam5996. This result confirms our findings with the simulated data, i.e., PROCLUS fails to select relevant features in high dimensions. In both cases, K-means and EM(d) are unable to discover the two groups in the data: almost all emails are clustered in a single group. In Figures 7-9 we plot the error rate of LAC as a function of the input parameter h for the three text datasets used in our experiments. As expected, the accuracy of the LAC algorithm is sensitive to the value of h; nevertheless, a good performance was achieved across the range of values tested ($\frac{1}{h} = 1, 3, 5, 7, 9, 11$).

We run the LAC and PROCLUS algorithms using the microarray data and small values of k ($k = 3$ and $k = 4$). Tables XV and XVI show sizes, scores, and dimensions of the biclusters detected by LAC and PROCLUS. For this dataset, DOC was not able to find any clusters. For LAC we have selected the dimensions with the largest weights ($1/h$ is fixed to 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 "labeled" 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 both Tables XV and XVI).

In general, the weighting of dimensions provides a convenient scheme to properly tune the results. That is: by ranking the dimensions according to their weight, we can keep adding to a cluster the dimension that minimizes the increase in score. Thus, given an upper bound on the score, we can obtain the largest set of dimensions that satisfies the given bound.

To assess the biological significance of generated clusters we used a biological data mart (developed by our collaborator biologists), that employs an agent framework to maintain knowledge from external databases. Significant themes were observed in some of these groups. For example, one cluster (shown in Table XVII) contains a number of cell cycle genes. The terms for cell cycle regulation all score high. As with all cancers, BRCA1- and BRCA2-related tumors involve the loss of control over cell growth and proliferation, thus the presence of strong cell-cycle components in the clustering is expected.

7. Conclusions and Future Work

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. This approach avoids the risk of loss of information encountered in global dimensionality reduction techniques.

The output of our algorithm is twofold. It provides a partition of the data, so that the points in each set of the partition constitute a cluster. In addition, each set is associated with a weight vector, whose values give information of the degree of relevance of features for each partition. Our experiments show that there is a perfect correspondence between the weight values of each cluster and local correlations of data.

We have formally proved that our algorithm converges to a local minimum of the associated error function, and experimentally demonstrated the gain in perfomance we achieve with our method in high

dimensional spaces with clusters folded in subspaces spanned by different combinations of features. In addition, we have shown the feasibility of our technique to discover "good" biclusters in microarray gene expression data.

The LAC algorithm performed extremely well on the three high dimensional text data (Classic3, Spam2000, and Spam5996). In our future work we will further investigate the use of LAC for keyword identification of unlabeled documents. An analysis of the weights credited to terms can guide the automatic identification of class-specific keywords, and thus the process of label assignment to clusters. These findings can have a relevant impact for the retrieval of information in content-based indexed documents.

The LAC algorithm requires as input parameter the value of h, which controls the strength of the incentive for clustering on more features. To generate robust and stable solutions, new consensus subspace clustering methods are under investigation by the authors. The major difficulty is to find a consensus partition from the output partitions of the contributing clusterings, so that an "improved" overall clustering of the data is achieved. Since we are dealing with weighted clusters, proper consensus functions that make use of the weight vectors associated with the clusters will be investigated.

Table VI. Confusion matrices for Example2.

LAC	*C0* (input)	*C1* (input)
C0 (output)	2486	13
C1 (output)	14	2487
PROCLUS	*C0* (input)	*C1* (input)
C0 (output)	1755	648
C1 (output)	745	1852
K-means	*C0* (input)	*C1* (input)
C0 (output)	1355	1273
C1 (output)	1145	1227

Table VII. Confusion matrices for Example3.

LAC	*C0* (input)	*C1* (input)
C0 (output)	2497	1
C1 (output)	3	2499
PROCLUS	*C0* (input)	*C1* (input)
C0 (output)	2098	676
C1 (output)	402	1824
K-means	*C0* (input)	*C1* (input)
C0 (output)	1267	1171
C1 (output)	1233	1329

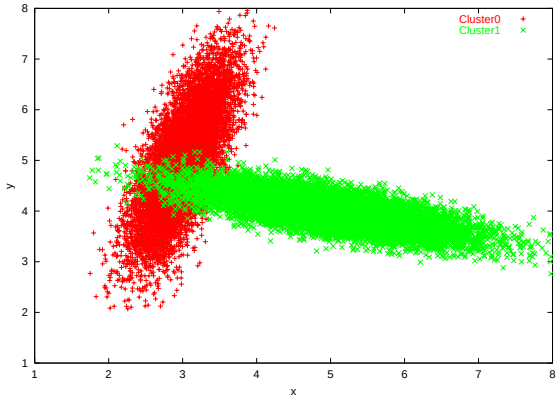

Figure 3. Example4: Two Gaussian clusters non-axis oriented in two dimensions.

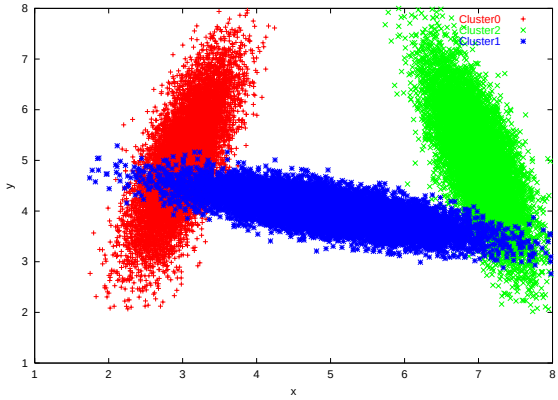

Figure 4. Example5: Three Gaussian clusters non-axis oriented in two dimensions.

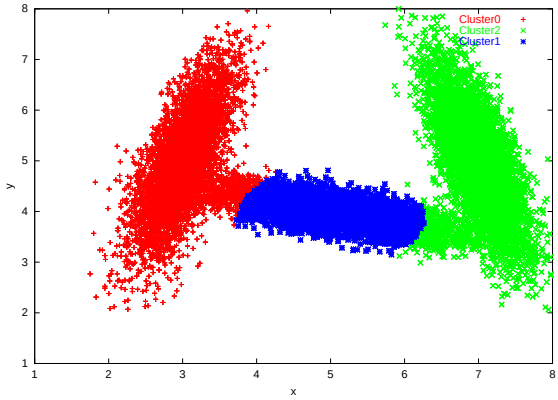

Figure 5. Example5: Clustering results of the LAC algorithm.

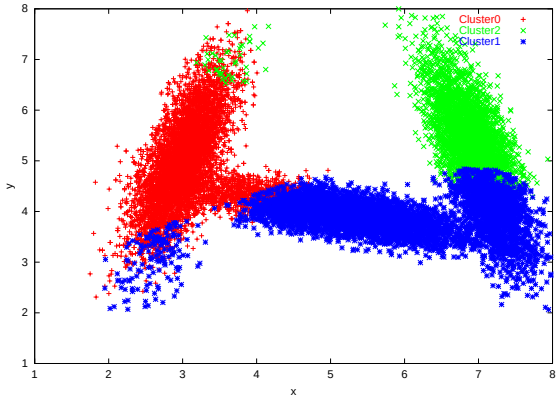

Figure 6. Example5: Clustering results of K-means.

Table VIII. Confusion matrices for Example4.

LAC	*C0* (input)	*C1* (input)
C0 (output)	4998	473
C1 (output)	2	4527
PROCLUS	*C0* (input)	*C1* (input)
C0 (output)	5000	714
C1 (output)	0	4286
K-means	*C0* (input)	*C1* (input)
C0 (output)	4956	724
C1 (output)	44	4276

Table IX. LAC: Weight values for Example4.

Cluster	w_1	w_2
C0	0.99	0.01
C1	0.09	0.91

Table X. Confusion matrices for Example5.

LAC	*C0* (input)	*C1* (input)	*C2* (input)
C0 (output)	5000	622	0
C1 (output)	0	3844	0
C2 (output)	0	534	5000
PROCLUS	*C0* (input)	*C1* (input)	*C2* (input)
C0 (output)	5000	712	0
C1 (output)	0	4072	117
C2 (output)	0	216	4883
K-means	*C0* (input)	*C1* (input)	*C2* (input)
C0 (output)	4816	1018	0
C1 (output)	140	3982	1607
C2 (output)	44	0	3393

Table XI. LAC: Weight values for Example5.

Cluster	w_1	w_2
C0	0.92	0.08
C1	0.44	0.56
C2	0.94	0.06

Table XII. Average error rates for real data.

	LAC	*PROCLUS*	*K-means*	*DOC*	*EM (d)*	*EM (f)*
OQ	**30.9**	31.6	47.1	54.0	40.0	43.8
Breast	**4.5**	5.7	**4.5**	32.9	5.3	5.4
Pima	29.6	33.1	**28.9**	42.7	33.7	34.9
Image	39.1	42.5	38.3	45.8	39.8	**34.6**
Sonar	**38.5**	39.9	46.6	65.0	44.5	44.3
Lymphoma	**32.3**	33.3	39.6	–	47.4	–
Classic3	**2.6**	48.2	62.4	–	59.2	–
Spam2000	**1.2**	28.0	44.7	–	36.6	–
Spam5996	**5.1**	44.5	44.9	–	44.8	–
Average	**20.4**	34.1	39.7	48.1	39.0	32.6

Table XIII. Confusion matrices for Spam2000.

LAC	*Spam* (input)	*Non-spam* (input)
Spam (output)	771	2
Non-spam (output)	15	640
PROCLUS	*Spam* (input)	*Non-spam* (input)
Spam (output)	502	116
Non-spam (output)	284	526
K-means	*Spam* (input)	*Non-spam* (input)
Spam (output)	786	639
Non-spam (output)	0	3
EM(d)	*Spam* (input)	*Non-spam* (input)
Spam (output)	781	517
Non-spam (output)	5	125

Table XIV. Confusion matrices for Spam5996.

LAC	*Spam* (input)	*Non-spam* (input)
Spam (output)	733	20
Non-spam (output)	53	622
PROCLUS	*Spam* (input)	*Non-spam* (input)
Spam (output)	777	627
Non-spam (output)	9	15
K-means	*Spam* (input)	*Non-spam* (input)
Spam (output)	786	641
Non-spam (output)	0	1
EM(d)	*Spam* (input)	*Non-spam* (input)
Spam (output)	780	634
Non-spam (output)	6	8

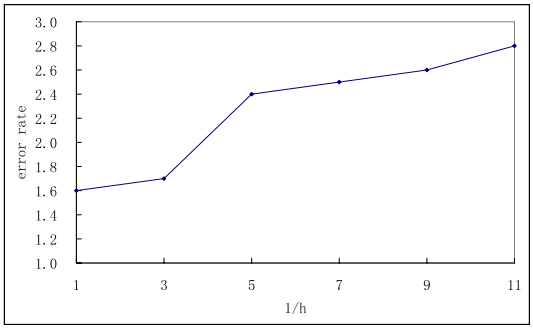

Figure 7. Classic3 dataset: Error rate of LAC versus $\frac{1}{h}$ parameter.

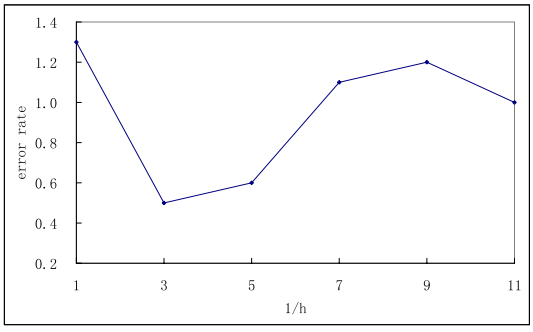

Figure 8. Spam2000 dataset: Error rate of LAC versus $\frac{1}{h}$ parameter.

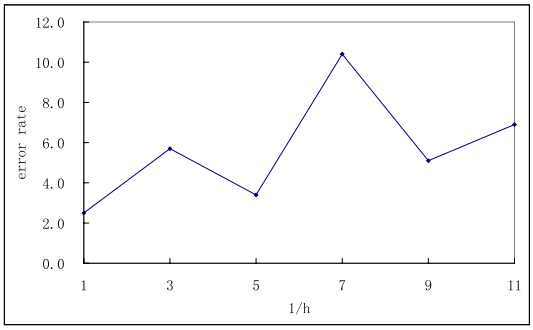

Figure 9. Spam5996 dataset: Error rate of LAC versus $\frac{1}{h}$ parameter.

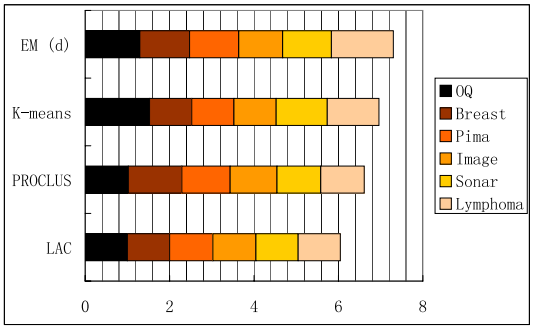

Figure 10. Performance distributions over real datasets.

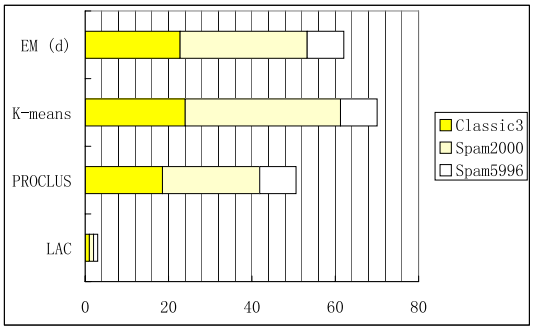

Figure 11. Performance distributions over text data.

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

$k = 3$	*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

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

$k = 4$	*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

Table XVII. Biological processes annotated in one cluster generated by the LAC algorithm.

Biological process	**z-score**	**Biological process**	**z-score**
DNA damage checkpoint	7.4	purine nucleotide biosynthesis	4.1
nucleocytoplasmic transport	7.4	mRNA splicing	4.1
meiotic recombination	7.4	cell cycle	3.5
asymmetric cytokinesis	7.4	negative regulation of cell proliferation	3.4
purine base biosynthesis	7.4	induction of apoptosis by intracellular signals	2.8
GMP biosynthesis	5.1	oncogenesis	2.6
rRNA processing	5.1	G1/S transition of mitotic cell cycle	2.5
glutamine metabolism	5.1	protein kinase cascade	2.5
establishment and/or maintenance of cell polarity	5.1	central nervous system development	4.4
gametogenesis	5.1	regulation of cell cycle	2.1
DNA replication	4.6	cell cycle arrest	4.4
glycogen metabolism	2.3		

References

Aggarwal, C., Procopiuc, C., Wolf, J. L., Yu, P. S., and Park, J. S. Fast Algorithms for Projected Clustering. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1999.

Aggarwal, and Yu, P. S. Finding generalized projected clusters in high dimensional spaces. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000.

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1998.

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

Arabie, P., and Hubert, L. J. An overview of combinatorial data analysis. Clustering and Classification. World Scientific Pub., pages 5–63, 1996.

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

Chakrabarti, K., and Mehrotra, S. Local dimensionality reduction: a new approach to indexing high dimensional spaces. *Proceedings of VLDB*, 2000.

Cheng, Y., and Church, G. M. Biclustering of expression data. *Proceedings of the eighth international conference on intelligent systems for molecular biology*, 2000.

Cheeseman, P., and Stutz, J. Bayesian classification (autoclass): theory and results. *Advances in Knowledge Discovery and Data Mining*, Chapter 6, pages 153–180, AAAI/MIT Press, 1996.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

Dhillon, I. S., Mallela, S., and Modha, D. S. Information-theoretic co-clustering. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

Domeniconi, C., Papadopoulos, D., Gunopulos, D., and Ma, S. Subspace Clustering of High Dimensional Data. *Proceedings of the SIAM International Conference on Data Mining*, 2004.

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

Dy, J. G., and Brodley, C. E. Feature Subset Selection and Order Identification for Unsupervised Learning. *Proceedings of the International Conference on Machine Learning*, 2000.

Ester, M., Kriegel, H. P., and Xu, X. A database interface for clustering in large spatial databases. *Proceedings of the 1st Int'l Conference on Knowledge Discovery in Databases and Data Mining*, 1995.

Friedman, J., and Meulman, J. Clustering Objects on Subsets of Attributes. *Technical Report, Stanford University*, September 2002.

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

Ghahramani, Z., and Hinton, G. E. The EM Algorithm for Mixtures of Factor Analyzers. *Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto*, 1996.

Hedenfalk, I., Duggan, D., Chen, Y., Radmacher, M., Bittner, M., Simon, R., Meltzer, P., Gusterson, B., Esteller, M., Kallioniemi, O. P., Wilfond, B., Borg,

A., and Trent, J. Gene expression profiles in hereditary breast cancer. *N Engl J Med*, 344:539–548, 2001.

Keogh, E., Chakrabarti, K., Mehrotra, S., and Pazzani, M. Locally adaptive dimensionality reduction for indexing large time series databases. *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2001.

Michalski, R. S., and Stepp, R. E. Learning from observation: Conceptual clustering. *Machine Learning: An Artificial Intelligence Approach*, in R. S. Michalski, J. G. Carbonell, and T. M. Mitchell editors, 1996.

Mladenović, N., and Brimberg, J. A degeneracy property in continuous location-allocation problems. *Les Cahiers du GERAD*, G-96-37, Montreal, Canada, 1996.

Modha, D., and Spangler, S. Feature Weighting in K-Means Clustering. *Machine Learning*, 52(3):217–237, 2003.

Ng, R. T., and Han, J. Efficient and effective clustering methods for spatial data mining. *Proceedings of the VLDB conference*, 1994.

Procopiuc, C. M., Jones, M., Agarwal, P. K., and Murali, T. M. A Monte Carlo algorithm for fast projective clustering. *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2002.

Tipping, M. E., and Bishop, C. M. Mixtures of Principal Component Analyzers. *Neural Computation*, 1(2):443–482, 1999.

Thomasian, A., Castelli, V., and Li, C. S. Clustering and singular value decomposition for approximate indexing in high dimensional spaces. *Proceedings of CIKM*, 1998.

Wang, H., Wang, W., Yang, J., and Yu, P. S. Clustering by pattern similarity in large data sets. *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2002.

Wu, C. F. J. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11(1):95–103, 1983.

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