Data Min Knowl Disc (2007) 14:63–97
DOI 10.1007/s10618-006-0060-8

Locally adaptive metrics for clustering high dimensional data

Carlotta Domeniconi · Dimitrios Gunopulos · Sheng Ma · Bojun Yan · Muna Al-Razgan · Dimitris Papadopoulos

Received: 11 January 2006 / Accepted: 20 November 2006 /
Published online: 26 January 2007
Springer Science+Business Media, LLC 2007

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 · Clustering ensembles · Gene expression data · Text data

Responsible editor: Johannes Gehrke.

C. Domeniconi (✉)· B. Yan · M. Al-Razgan
George Mason University, Fairfax, VA, USA
e-mail: carlotta@ise.gmu.edu

D. Gunopulos · D. Papadopoulos
UC Riverside, Riverside, CA, USA

S. Ma
Vivido Media Inc., Suite 319, Digital Media Tower 7 Information Rd.,
Shangdi Development Zone,
Beijing 100085, China

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 1983), 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 multidimensional 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 upper plot depicts two clusters of data elongated along the x and y dimensions. The lower 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,

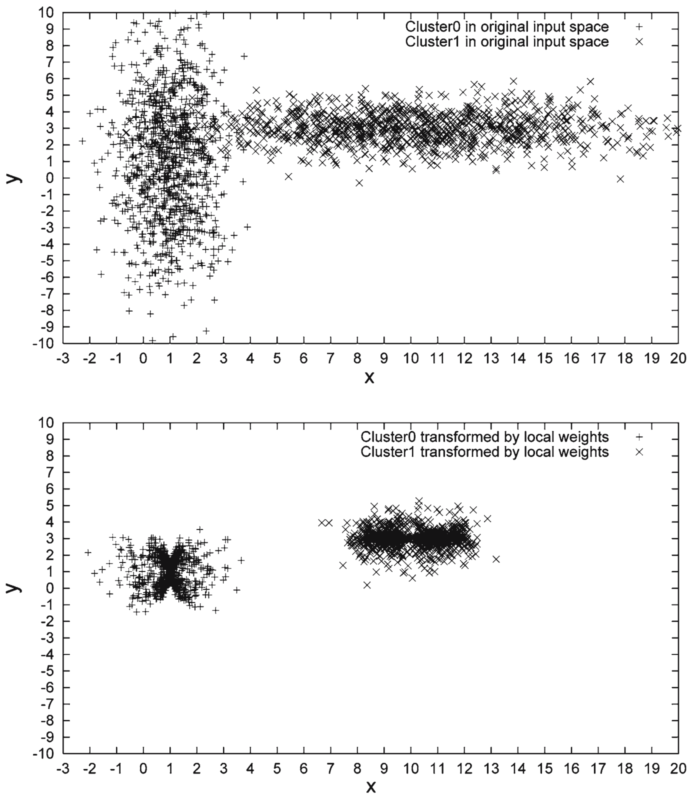

Fig. 1 (*Top*) Clusters in original input space. (*Bottom*) Clusters transformed by local weights

and allows for the discovery of different patterns in different subspaces of the original input space.

1.1 Our contribution

An earlier version of this work appeared in (Domeniconi et al. 2004; Al-Razgan and Domeniconi 2006). 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.
4. The LAC algorithm requires in input a parameter that controls the strength of the incentive for clustering on multiple dimensions. The setting of such parameter is particularly difficult, since no domain knowledge for its tuning is available. In this paper, we introduce an ensemble approach to combine multiple weighted clusters discovered by LAC using different input parameter values. The result is a consensus clustering that is superior to the participating ones, provided that the input clusterings are diverse. Our ensemble approach gives a solution to the difficult and crucial issue of tuning the input parameter of LAC.

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. PCA 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.

One of the earliest work that discusses the problem of clustering simultaneously both points and dimensions is (Hartigan 1972). A model based on direct clustering of the data matrix and a distance-based model are introduced, both leading to similar results.

More recently, 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 Fig. 1 (Top), 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 does not 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 contrast 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 random 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. Parsons et al. (2004) provides a good overview of subspace clustering techniques for high-dimensional data.

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) and 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) and 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 COSA

Our technique LAC is related to the algorithm Clustering On Subsets of Attributes (COSA), proposed in Friedman and Meulman (2002). Both LAC and COSA develop an exponential weighting scheme, but they are fundamentally different in their search strategies and their outputs. COSA is an iterative algorithm that assigns a weight vector (with a component for each dimension) to *each data point*, while LAC assigns a weight vector to *each cluster* instead.

The COSA starts by assigning equal weight values to each dimension and to all points. It then considers the k nearest neighbors of each point, and uses the resulting neighborhoods to compute the dimension weights. Larger weights are credited to those dimensions that have a smaller dispersion within the neighborhood. These weights are then used to compute dimension weights for each pair of points, which in turn are utilized to update the distances for the computation of the k nearest neighbors. The process is iterated until the weight values become stable.

At each iteration, the neighborhood of each point becomes increasingly populated with data from the same cluster. The final output is a pairwise distance matrix based on a weighted inverse exponential distance that can be used as input to any distance-based clustering method (e.g., hierarchical clustering).

The COSA requires in input the value of k for nearest neighbor computation. COSA does not require in input the number of dimensions per cluster. As in LAC, this value is regulated by a parameter (called h in this paper) that controls the strength of the incentive for clustering on multiple dimensions. COSA requires the tuning of such parameter. The setting of h is particularly difficult, since no domain knowledge for its tuning is available. As a major advantage with respect to COSA, in this paper, we introduce an ensemble approach to combine multiple weighted clusters discovered by LAC using different h values. The result is a consensus clustering that is superior to the participating ones, and resolves the issue of tuning the parameter h. The details of our ensemble approach are presented in Sect. 6.

2.2 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 sets 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). Similarly, Yang et al. (2002) introduces the δ-cluster model to discover strong coherence among a set of objects (on a subset of dimensions), even if they have quite different values, and the dimension values are not fully specified. The concept of bicluster (Cheng and Church 2000) (which assumes that the microarray matrix is fully specified) can be regarded as a special case of this model.

3 Locally adaptive metrics for clustering

We define what we call *weighted cluster*. Consider a set of points in some space of dimensionality D. A *weighted cluster* C is a subset of data points, together with a vector of weights $\mathbf{w} = (w_1, \ldots, w_D)$, 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 participation of feature j to the cluster C. If the points in C are well clustered along feature j, w_j is large, otherwise it is small. 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 N points $\mathbf{x}$ in the D-dimensional Euclidean space, a set of k centers $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$, $\mathbf{c}_j \in \Re^D, j = 1, \ldots, k$, coupled with a set of corresponding weight vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$, $\mathbf{w}_j \in \Re^D, j = 1, \ldots, k$, partition S into k sets $\{S_1, \ldots, S_k\}$:

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

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:

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

subject to the constraints $\sum_i w_{ji} = 1 \; \forall j$. C and W are $(D \times k)$ 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} is the variance of the data 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 variance 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 variance of data within each cluster. One way to achieve this goal is to add the regularization term $\sum_{i=1}^{D} w_{ji} \log w_{ji}$,[1] which represents the negative entropy of the weight distribution for each cluster (Friedman and Meulman 2002). It penalizes solutions with maximal (unit) weight on the single feature with smallest variance within each cluster. The resulting error function is

$$E_2(C, W) = \sum_{j=1}^{k} \sum_{i=1}^{D} (w_{ji} X_{ji} + h w_{ji} \log w_{ji}) \tag{4}$$

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

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

problem by introducing the Lagrange multipliers λ_j (one for each constraint), and minimizing the resulting (unconstrained now) error function

$$E(C, W) = \sum_{j=1}^{k} \sum_{i=1}^{D} (w_{ji} X_{ji} + h w_{ji} \log w_{ji}) + \sum_{j=1}^{k} \lambda_j \left(1 - \sum_{i=1}^{D} w_{ji} \right). \tag{5}$$

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:

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

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

Solving Eq. 6 with respect to w_{ji} we obtain $h \log w_{ji} = -X_{ji} + \lambda_j - h$. 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 Eq. 7:

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

Solving with respect to λ_j we obtain

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

Thus, the optimal w_{ji}^* is

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

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:

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

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 variance 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 $E_0(C) = \frac{1}{D}\sum_{j=1}^{k}\sum_{i=1}^{D} X_{ji}$, 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 $1/D$. Given the initial centroids $\mathbf{c}_j$, for $j = 1,\dots,k$, 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 Eq. 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 Eq. 9. The procedure is iterated until convergence is reached. The resulting algorithm, that we call LAC, is summarized in the following.

```
Input N points $\mathbf{x} \in R^D$, $k$, and $h$.

1. Start with $k$ initial centroids $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$;
2. Set $w_{ji} = 1/D$, for each centroid $\mathbf{c}_j$, $j = 1, \ldots, k$ and each feature $i = 1, \ldots, D$;
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 $L_w(\mathbf{c}_l, \mathbf{x}) = (\sum_{i=1}^{D} w_{li}(c_{li} - x_i)^2)^{1/2}$;
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 $w_{ji} = \frac{\exp(-X_{ji}/h)}{\sum_{l=1}^{D} \exp(-X_{jl}/h)}$;
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 $O(kDN)$.

Despite the similarities between the LAC algorithm and EM with a diagonal covariance matrix, our approach has distinctive characteristics that make it different from the EM algorithm. Specifically: (1) LAC performs a hard assignment of points to clusters. (2) LAC assumes equal prior probabilities for all clusters, as any centroid-based clustering approach does. (3) The variance X_{ji}, for each cluster j and each dimension i, is estimated directly from the data in a nonparametric fashion, without assuming any data distribution model. (4) The exponential weighting scheme provided by Eq. 8 results in a faster rate of convergence (as corroborated by our experimental results). In fact, variations of clusters' variances X_{ji} are exponentially reflected into the corresponding weight values w_{ji}. Thus, the weights are particularly sensitive to changes in local feature relevance. (5) The weights w_{ji} credited to each cluster j and to each dimension i are determined by both a local and a global component: X_{ji} and h, respectively. X_{ji} is the variance of the data along dimension i, within cluster j. The constraint $\sum_i w_{ji} = 1$ makes the weights to measure the proportional amounts of the dimensions that account for the variances of the clusters. The global parameter h is equal for all clusters and all dimensions. It controls how much the distribution of weight values will deviate from the uniform distribution.

We point out that the LAC algorithm can identify a degenerate solution, i.e., a partition with empty clusters, during any iteration. Although we did not 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 nonempty 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 nondegenerate 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 nondegenerate solution.

5 Convergence of the LAC algorithm

In light of the remark made above on the analogy of LAC with the dynamics of 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 Eqs. 8 and 9, steps 4 and 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 Eqs. 10 and 11, respectively. The value of the error function E at completion of iteration t is $E_5^{(t)}(C', W')$, where we explicit the dependence of E on the partition of points computed in step 5 of the algorithm. C' and W' are the matrices of the newly computed centroids and weights. Since the new partition computed in step 3 of the successive iteration $t+1$ 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:

$$E_3^{(t+1)}(C', W') - E_5^{(t)}(C', W') \leq 0, \tag{12}$$

where $E_3^{(t+1)}$ is the error function evaluated on the partition computed in step 3 of the successive iteration $t+1$. Using this result, and the identities $E(C', W') = E_3^{(t+1)}(C', W')$ and $E(C, W) = E_3^{(t)}(C, W)$ [$E_3^{(t)}$ is the value of the error function at the beginning (step 3) of iteration t], we can derive the following inequality:

$$\begin{aligned} E(C', W') - E(C, W) &= E_3^{(t+1)}(C', W') - E_5^{(t)}(C', W') \\ &\quad + E_5^{(t)}(C', W') - E_3^{(t)}(C, W) \\ &\leq E_5^{(t)}(C', W') - E_3^{(t)}(C, W) \leq 0, \end{aligned}$$

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 Setting parameter *h*: an ensemble approach

The clustering result of LAC depends on two input parameters. The first one is common to all clustering algorithms: the number of clusters k to be discovered in the data. The second one (h) controls the strength of the incentive to cluster on more features. The setting of h is particularly difficult, since no domain knowledge for its tuning is available. Here we focus on setting the parameter h directly from the data. We leverage the diversity of the clusterings produced by LAC when different values of h are used, in order to generate a consensus clustering that is superior to the participating ones. The major challenge we face is to find a consensus partition from the outputs of the LAC algorithm to achieve an "improved" overall clustering of the data. Since we are dealing with weighted clusters, we need to design a proper consensus function that makes use of the weight vectors associated with the clusters. Our techniques leverage such weights to define a similarity measure which is associated to the edges of a bipartite graph. The problem of finding a consensus function is then mapped to a graph partitioning problem (Dhillon 2001; Fern and Brodley 2004; Strehl and Ghosh 2003).

The LAC algorithm outputs a partition of the data, identified by the two sets $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ and $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$. Our aim here is to generate robust and stable solutions via a consensus clustering method. We can generate contributing clusterings by changing the parameter h. The objective is then to find a consensus partition from the output partitions of the contributing clusterings, so that an "improved" overall clustering of the data is obtained.

For each data point $\mathbf{x}_i$, the weighted distance from cluster C_l is given by

$$d_{il} = \sqrt{\sum_{s=1}^{D} w_{ls}(x_{is} - c_{ls})^2}.$$

Let $D_i = \max_l\{d_{il}\}$ be the largest distance of $\mathbf{x}_i$ from any cluster. We want to define the probability associated with cluster C_l given that we have observed $\mathbf{x}_i$. At a given point $\mathbf{x}_i$, the cluster label C_l is assumed to be a random variable from a distribution with probabilities $\{P(C_l|\mathbf{x}_i)\}_{l=1}^{k}$. We provide a nonparametric estimation of such probabilities based on the data and on the clustering result.

In order to embed the clustering result in our probability estimations, the smaller the distance d_{il} is, the larger the corresponding probability credited to C_l should be. Thus, we can define $P(C_l|\mathbf{x}_i)$ as follows:

$$P(C_l|\mathbf{x}_i) = \frac{D_i - d_{il} + 1}{kD_i + k - \sum_l d_{il}}, \tag{13}$$

where the denominator serves as a normalization factor to guarantee $\sum_{l=1}^{k} P(C_l|\mathbf{x}_i) = 1$. We observe that $\forall l = 1, \ldots, k$ and $\forall i = 1, \ldots, N$ $P(C_l|\mathbf{x}_i) > 0$. In particular, the added value of 1 in (13) allows for a nonzero probability $P(C_L|\mathbf{x}_i)$ when $L = \arg\max_l\{d_{il}\}$. In this last case, $P(C_l|\mathbf{x}_i)$ assumes its minimum value $P(C_L|\mathbf{x}_i) = 1/(kD_i + k + \sum_l d_{il})$. For smaller distance values d_{il}, $P(C_l|\mathbf{x}_i)$ increases proportionally to the difference $D_i - d_{il}$: the larger the deviation of d_{il} from D_i, the larger the increase. As a consequence, the corresponding cluster C_l becomes more likely, as it is reasonable to expect based on the information provided by the clustering process. Thus, Eq. 13 provides a nonparametric estimation of the posterior probability associated to each cluster C_l.

We can now construct the vector P_i of posterior probabilities associated with $\mathbf{x}_i$:

$$P_i = (P(C_1|\mathbf{x}_i), P(C_2|\mathbf{x}_i), \ldots, P(C_k|\mathbf{x}_i))^t, \tag{14}$$

where t denotes the transpose of a vector. The transformation $\mathbf{x}_i \rightarrow P_i$ maps the D-dimensional data points $\mathbf{x}_i$ onto a new space of *relative coordinates* with respect to cluster centroids, where each dimension corresponds to one cluster. This new representation embeds information from both the original input data and the clustering result.

Suppose we run LAC m times for different values of the h parameter. For each point $\mathbf{x}_i$, and for each clustering $\nu = 1, \ldots, m$ we then can compute the vector of posterior probability P_i^ν. Using the P vectors, we construct the following matrix A:

$$A = \begin{pmatrix} (P_1^1)^t & (P_1^2)^t & \dots & (P_1^m)^t \\ (P_2^1)^t & (P_2^2)^t & \dots & (P_2^m)^t \\ \vdots & \vdots & & \vdots \\ (P_N^1)^t & (P_N^2)^t & \dots & (P_N^m)^t \end{pmatrix}.$$

Note that the $(P_i^\nu)^t$s are row vectors (t denotes the transpose). The dimensionality of A is therefore $N \times km$, under the assumption that each of the m clusterings produces k clusters. (We observe that the definition of A can be easily generalized to the case where each clustering may discover a different number of clusters.)

Based on A we can now define a bipartite graph to which our consensus partition problem maps. Consider the graph $G = (V, E)$ with V and E constructed as follows. $V = V^C \cup V^I$, where V^C contains km vertices, each representing a cluster of the ensemble, and V^I contains N vertices, each representing an input data point. Thus, $|V| = km + N$. The edge E_{ij} connecting the vertices V_i and V_j is assigned a weight value defined as follows. If the vertices V_i and V_j represent both clusters or both instances, then $E(i,j) = 0$; otherwise, if vertex V_i represents an instance $\mathbf{x}_i$ and vertex V_j represents a cluster C_j^ν (or vice versa) then the corresponding entry of E is $A(i, k(\nu - 1) + j)$.

We note that the dimensionality of E is $(km + N) \times (km + N)$, and E can be written as follows:

$$E = \begin{pmatrix} 0 & A^t \\ A & 0 \end{pmatrix}.$$

A partition of the bipartite graph G partitions the cluster vertices and the instance vertices simultaneously. The partition of the instances can then be output as the final clustering. Due to the special structure of the graph G (sparse graph), the size of the resulting bipartite graph partitioning problem is kmN. We run METIS (Kharypis and Kumar 1995) on the resulting bipartite graph to compute a k-way partitioning that minimizes the edge weight-cut. This gives the consensus clustering we seek. We call the resulting algorithm Weighted Bipartite Partitioning Algorithm (WBPA) . An earlier version of this algorithm appeared in Al-Razgan and Domeniconi (2006).

7 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 (with Euclidean distance), MK-means (K-means with Mahalanobis distance), 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. Furthermore, we include a comparison with a clustering method which puts unit (i.e., maximal) weight on a single dimension. In order to estimate the most relevant dimension for each cluster, we apply LAC, and consider the dimension that receives the largest weight within each cluster. Data within each cluster are then projected along the corresponding selected dimension. This gives a one-dimensional centroid per cluster. Finally, data are partitioned as in K-means using the resulting one-dimensional distances (each cluster uses, in general, a different dimension). We call the resulting algorithm LAC(1-dim).

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.

7.1 Simulated data

Example 1 The dataset consists of $D = 2$ 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 2 shows the results for this problem. We generated 60,000 data points, and performed ten fold cross-validation with 30,000 training data and 30,000 testing data.

Example 2 This dataset consists of $D = 30$ 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 2 shows the results for this problem. We generated 10,000 data points, and performed ten fold cross-validation with 5,000 training and 5,000 testing data.

Example 3 This dataset consists of $D = 50$ input features and $k = 2$ clusters. Both clusters are distributed according to multivariate Gaussians. Mean

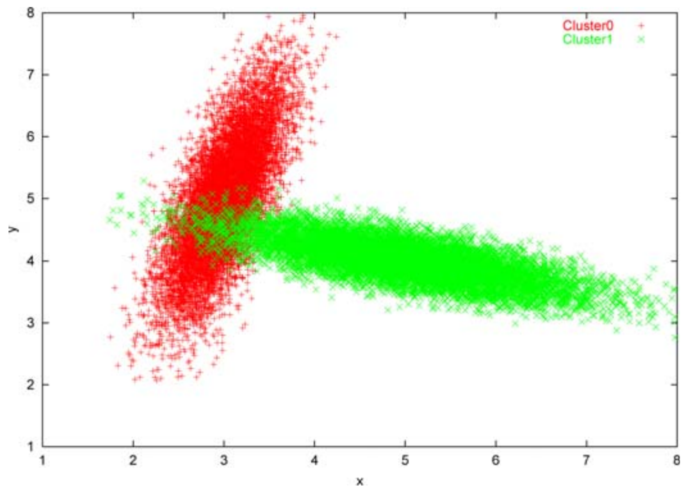

Fig. 2 Example 4: two Gaussian clusters non-axis oriented in two dimensions

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 2 shows the results for this problem. We generated 10,000 data points, and performed ten fold cross-validation with 5,000 training data and 5,000 testing data.

Example 4 This dataset consists of off-axis oriented clusters, with $D = 2$ and $k = 2$. Figure 2 shows the distribution of the points for this dataset. We generated 20,000 data points, and performed five fold-cross-validation with 10,000 training data and 10,000 testing data. Table 2 shows the results.

Example 5 This dataset consists again of off-axis oriented two dimensional clusters. This dataset contains three clusters, as Fig. 3 depicts. We generated 30,000 data points, and performed five fold-cross-validation with 15,000 training data and 15,000 testing data. Table 2 shows the results.

7.2 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),

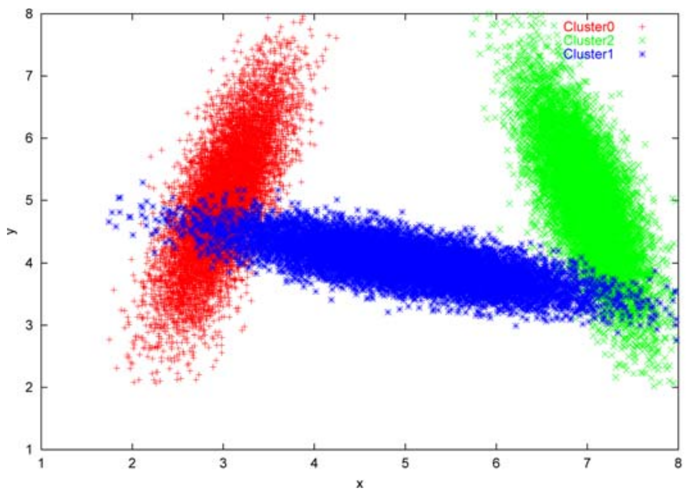

Fig. 3 Example 5: three Gaussian clusters non-axis oriented in two dimensions

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 2,000 terms (Spam2000), in the second case to 5,996 (Spam5996). In both cases, we consider two clusters by merging the conference and jobs mails into one group (non-spam).

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 4,026 expression values. We clustered the samples with the expression values of the genes as attributes (4,026 dimensions). The samples are categorized into nine 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 3,226 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.

Biologists are interested in finding sets 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

Table 1 Characteristics of real data

	N	*D*	*k*	Number of data per cluster
OQ	1,536	16	2	783-753
Breast	683	9	2	444-239
Pima	768	8	2	500-268
Image	640	16	15	80-64-64-64-48-48-48-32-32-32-32-32-32-16-16
Sonar	208	60	2	111-97
Lymphoma	96	4,026	9	46-11-10-9-6-6-4-2-2
Classic3	3,893	3,302	3	1460-1400-1033
Spam2000	1,428	2,000	2	786-642
Spam5996	1,428	5,996	2	786-642
DNA-microarray	3,226	22	n.a.	n.a.

Table 2 Average error rates for simulated data

	LAC	LAC(1-dim)	PROCLUS	*K*-means	MK-means	DOC	EM(d)	EM(f)
Ex1	11.4(0.3)	13.3(0.3)	13.8(0.7)	24.2(0.5)	18.6(7.2)	35.2(2.2)	**5.1**(0.4)	**5.1**(0.4)
Ex2	**0.5**(0.4)	26.5(2.2)	27.9(9.8)	48.4(1.1)	42.3(7.7)	no clusters	0.6(0.3)	0.8(0.3)
Ex3	0.08(0.1)	28.6(2.3)	21.6(5.3)	48.1(1.1)	47.7(1.5)	no clusters	**0.0**(0.1)	25.5(0.2)
Ex4	4.8(0.4)	4.9(0.4)	7.1(0.7)	7.7(0.7)	3.8(4.7)	22.7(5.9)	4.8(0.2)	**2.3**(0.2)
Ex5	7.7(0.3)	9.4(0.3)	7.0(2.0)	18.7(2.7)	4.8(4.9)	16.5(3.9)	6.0(0.2)	**2.3**(0.2)
Average	4.9	16.5	15.5	29.4	23.4	24.8	**3.3**	7.2

residue score (in general, below a certain threshold). The characteristics of all ten real datasets are summarized in Table 1.

7.3 Results on simulated data

The performance results reported in Table 2 clearly demonstrate the large gain in performance obtained by the LAC algorithm with respect to LAC(1-dim), PROCLUS, *K*-means and MK-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. Figure 4 shows the result obtained with LAC 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. Also MK-means gives large error rates on Examples 2 and 3, which demonstrates the difficulty of estimating full covariance matrices in high-dimensional spaces. As expected, projecting the data on one-dimension causes the loss of crucial information for the high-dimensional clusters of Examples 2 and 3. As a consequence, LAC(1-dim) performs considerably worse than LAC on these data. On the other hand, LAC(1-dim) performs reasonably well on Examples 1, 4, and 5, since the two-dimensional Gaussian data are designed to cluster closely along a single direction.

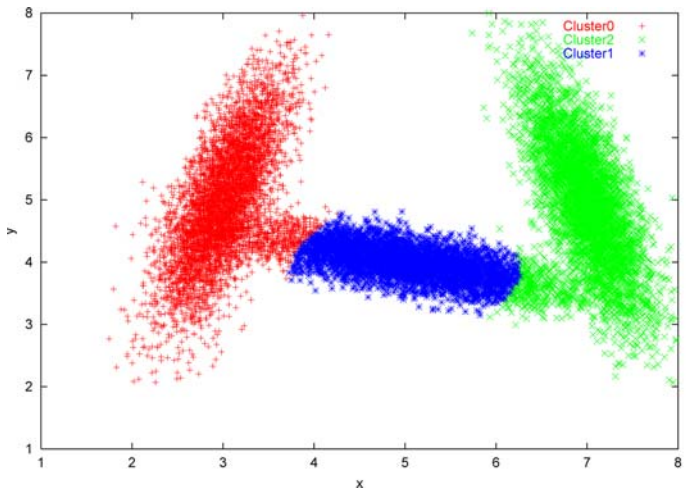

Fig. 4 Example 5: clustering results of the LAC algorithm

Table 3 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

Examples 1, 2, and 3 offer optimal conditions for EM(d); Examples 4 and 5 are optimal for EM(f) and MK-means. As a consequence, EM(d) and EM(f) provide best error rates in such respective cases. As expected, MK-means provides error rates similar to EM(f) on Examples 4 and 5. 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 confirms 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 2. 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 3 shows the dimensions selected by PROCLUS for each dataset and each cluster. Figure 5 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 5 shows that the performance of PROCLUS is highly sensitive to the value of its input

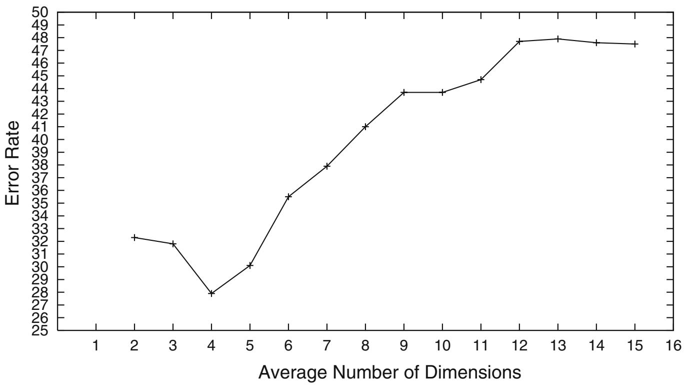

Fig. 5 Example 2: error rate of PROCLUS versus average number of dimensions

Table 4 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

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 4 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 convergence 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, N is the number of data points, and D 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 Eq. 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. Results are reported in Tables 5-9. We also computed the average weight values per cluster obtained over the runs conducted in our experiments. We report the weight values for Example 5 in Table 10. Similar results were obtained in all cases. Table 10 shows 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.

7.4 Results on real data

Table 11 reports the error rates obtained on the nine real datasets with class labels. For each dataset, all N points were used to identify the clusters. 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 D for each dataset, and report the best error rate obtained in

Table 5 Confusion matrices for Example 1

	$C0$ (input)	$C1$ (input)	$C2$ (input)
LAC			
$C0$ (output)	8,315	0	15
$C1$ (output)	1,676	10,000	1,712
$C2$ (output)	9	0	8,273
PROCLUS			
$C0$ (output)	7,938	0	7
$C1$ (output)	2,057	10,000	2,066
$C2$ (output)	5	0	7,927
K-means			
$C0$ (output)	9440	4,686	400
$C1$ (output)	411	3,953	266
$C2$ (output)	149	1,361	9,334

Table 6 Confusion matrices for Example 2

	$C0$ (input)	$C1$ (input)
LAC		
$C0$ (output)	2,486	13
$C1$ (output)	14	2,487
PROCLUS		
$C0$ (output)	1,755	648
$C1$ (output)	745	1,852
K-means		
$C0$ (output)	1,355	1,273
$C1$ (output)	1,145	1,227

Table 7 Confusion matrices for Example 3

	$C0$ (input)	$C1$ (input)
LAC		
$C0$ (output)	2,497	1
$C1$ (output)	3	2,499
PROCLUS		
$C0$ (output)	2,098	676
$C1$ (output)	402	1,824
K-means		
$C0$ (output)	1,267	1,171
$C1$ (output)	1,233	1,329

Table 8 Confusion matrices for Example 4

	$C0$ (input)	$C1$ (input)
LAC		
$C0$ (output)	4,998	473
$C1$ (output)	2	4,527
PROCLUS		
$C0$ (output)	5,000	714
$C1$ (output)	0	4,286
K-means		
$C0$ (output)	4,956	724
$C1$ (output)	44	4,276

each case. For the Lymphoma (4,026 dimensions), Classic3 (3,302 dimensions), Spam2000 (2,000 dimensions), and Spam5996 (5,996 dimensions) we tested several input parameter values of PROCLUS, and found the best result at 3,500, 350, 170, and 300, respectively. LAC gives the best error rate in six of nine datasets. LAC outperforms PROCLUS and EM(d) in each dataset. MK-means and EM do not perform well in general, and particularly in higher dimensions [the same holds for LAC(1-dim)]. This is likely due to the non-Gaussian distributions of real data. MK-means and EM(f) (Netlab library for Matlab) failed to run to completion on the very high-dimensional data due to memory prob-

Table 9 Confusion matrices for Example 5

	*C*0 (input)	*C1* (input)	*C2* (input)
LAC			
*C*0 (output)	5,000	622	0
*C*1 (output)	0	3844	0
*C*2 (output)	0	534	5,000
PROCLUS			
*C*0 (output)	5,000	712	0
*C*1 (output)	0	4,072	117
*C*2 (output)	0	216	4,883
K-means			
*C*0 (output)	4,816	1,018	0
*C*1 (output)	140	3,982	1,607
*C*2 (output)	44	0	3,393

Table 10 LAC: weight values for Example 5

Cluster	w_1	w_2
*C*0	0.92	0.08
*C*1	0.44	0.56
*C*2	0.94	0.06

Table 11 Average error rates for real data

	LAC	LAC(1-dim)	PROCLUS	*K*-means	MK-means	DOC	EM(*d*)	EM (*f*)
OQ	**30.9**	35.9	31.6	47.1	42.4	54.0	40.0	43.8
Breast	**4.5**	18.4	5.7	**4.5**	34.6	32.9	5.3	5.4
Pima	29.6	34.9	33.1	**28.9**	34.9	42.7	33.7	34.9
Image	39.1	54.5	42.5	38.3	35.4	45.8	39.8	**34.6**
Sonar	38.5	**33.2**	39.9	46.6	41.4	65.0	44.5	44.3
Lymphoma	**32.3**	52.1	33.3	39.6	–	–	47.4	–
Classic3	**2.6**	62.5	48.2	62.4	–	–	59.2	–
Spam2000	**1.2**	44.9	28.0	44.7	–	–	36.6	–
Spam5996	**5.1**	10.2	44.5	44.9	–	–	44.8	–
Average	**20.4**	38.5	34.1	39.7	37.7	48.1	39.0	32.6

lems. Interestingly, LAC(1-dim) gives the best error rate on the Sonar data, suggesting the presence of many noisy features. In three cases (Breast, Pima, and 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. Furthermore, the poor performance of LAC(1-dim) on the high-dimensional data (in particular, Lymphoma, Classic3, and Spam2000) demonstrates the presence of *multidimensional* sub-clusters embedded in very high-dimensional spaces. The discovery of one-dimensional clusters is not sufficient to reveal the underlying data structure.

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 11. 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. Nevertheless, 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.

7.4.1 Robustness analysis

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: $b_m = e_m / \min_{1 \le k \le 4} e_k$. 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 6 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 Fig. 7. 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, Fig. 7 graphically depicts the strikingly superior performance of LAC over the text data with respect to the competitive techniques.

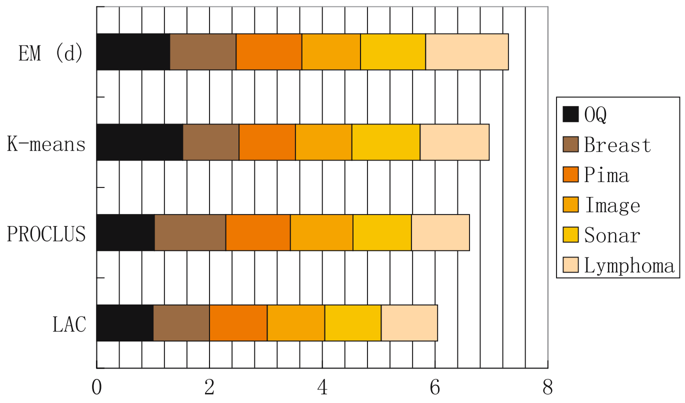

Fig. 6 Performance distributions over real datasets

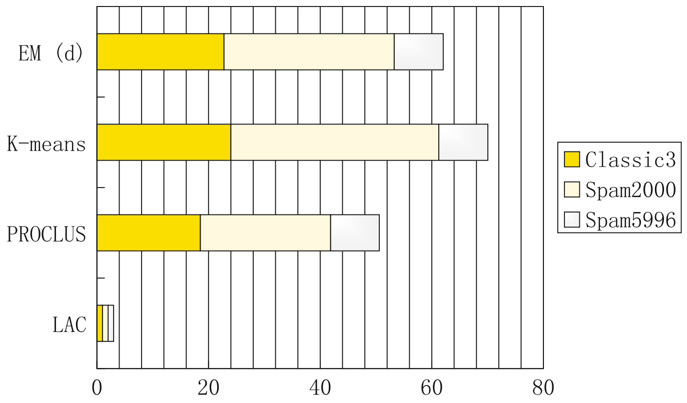

Fig. 7 Performance distributions over text data

7.4.2 Analysis of text data

To investigate the false positive and false negative rates on the spam data we show the corresponding confusion matrices in Tables 12 and 13. 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 Figs. 8–10, we plot the error rate of LAC

Table 12 Confusion matrices for Spam2000

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

Table 13 Confusion matrices for Spam5996

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

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 ($\frac{1}{h} = 1, 3, 5, 7, 9, 11$).

7.4.3 Analysis of microarray data

We ran the LAC and PROCLUS algorithms using the microarray data and small values of k ($k = 3$ and 4). Tables 14 and 15 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 ran PROCLUS with five as value of its input parameter. For $k = 4$, within two clusters five conditions receive again considerably larger weight than the

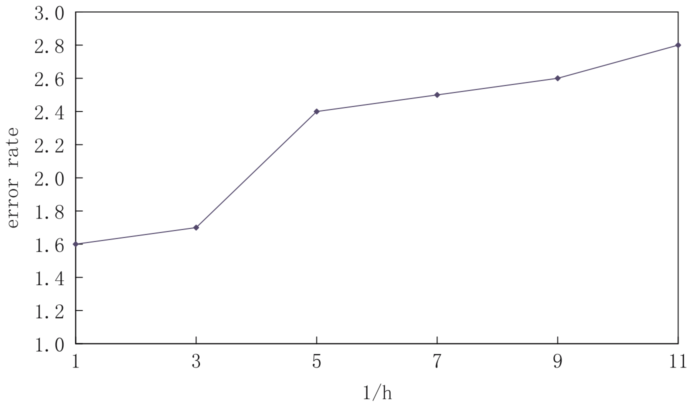

Fig. 8 Classic3 dataset: error rate of LAC versus $\frac{1}{h}$ parameter

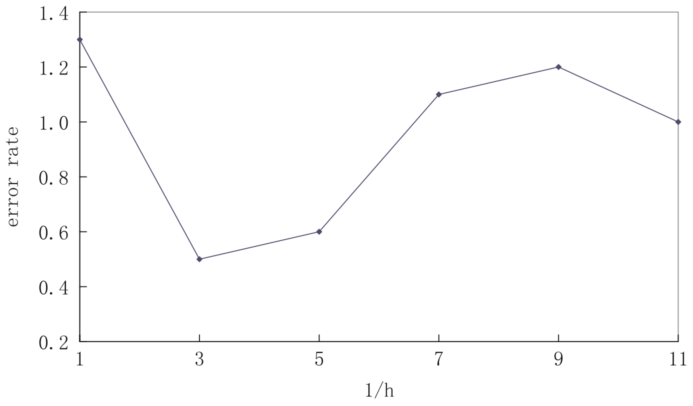

Fig. 9 Spam2000 dataset: error rate of LAC versus $\frac{1}{h}$ parameter

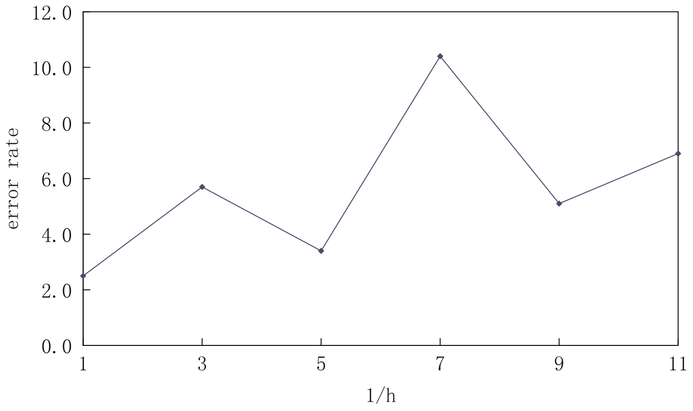

Fig. 10 Spam5996 dataset: error rate of LAC versus $\frac{1}{h}$ parameter

Table 14 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 15 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

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 14 and 15).

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 (corresponding to cluster $C1$ in Table 14) contains a number of cell cycle genes (see Table 16). 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.

Table 16 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		

Table 17 Error rates of WBPA

	WBPA	Min-Error	Max-Error	Avg-Error
OQ	47.5 (1.3)	31.3	49.0	48.3
Breast	3.6 (0.2)	5.9	34.1	20.5
Pima	31.9 (2.2)	29.2	33.6	30.6
Sonar	29.8 (1.7)	34.1	46.6	38.6
Classic3	2.2 (0.2)	1.9	33.8	9.3
Spam2000	0.70 (0.2)	0.62	1.5	0.97
Spam5996	1.2 (0.1)	1.9	7.0	3.8

7.4.4 Results on clustering ensembles

We ran the clustering ensemble algorithm WBPA on the real datasets described in Table 1. Since METIS (Kharypis and Kumar 1995) requires balanced datasets, we performed random sampling on Breast, Pima, Classic3, Spam2000, and Spam 5996. In each case, we sub-sampled the most populated class: from 444 to 239 for Breast, from 500 to 268 for Pima, from 1,460 and 1,400 to 1,033 for Classic3, and from 786 to 642 for Spam2000 and Spam5996. We did not use the Image and Lymphoma datasets to test the clustering ensemble technique since they are highly unbalanced and the smallest clusters contain too few points.

For each dataset, we ran the LAC algorithm for ten values of the input parameter $\frac{1}{h}$, randomly chosen within the set of values $\{0.1, 0.2, 0.5, 1, 2, 3, \ldots, 20\}$. The clustering results of LAC are then given in input to the consensus clustering technique WBPA. For the value of k, we input both LAC and the ensemble algorithm with the actual number of classes in the data. Results are provided in Table 17, where we report the error rates of WBPA (along with the corresponding standard deviations computed over five runs of WBPA), and the maximum, minimum, and average error rate values for the input clusterings.

The discrepancy between the Min-Error and Max-Error values show the sensitivity of the LAC algorithm on the value of the input parameter h. As the error rates in Table 17 demonstrate, the WBPA technique is capable of leveraging the diversity of the input clusterings, and provides a consensus clustering that is better or close to the best individual input clustering for almost all datasets. Specifically, for the Breast, Sonar, and Spam5996 datasets, the error rate of WBPA is lower than the best input error rate. For Classic3 and Spam2000, the error rate of WBPA is still very close to the best input error rate, and well below the average input error rate. For the Pima dataset, the WBPA error rate is slightly above the average input error rate. We observe that in this case the input error rates have a small variance, while the ensemble clustering technique is most effective when the input clusterings are diverse. WBPA does not perform well on the OQ dataset. The reason is that, among the input error rates, one has the value of 31.3%, while all the others are in the range 47–49%. LAC's accuracy and diversity are in this case too low for the ensemble to provide good results. It is well known, in fact, that a good tradeoff between diversity and accuracy is a necessary condition to achieve an effective ensemble approach. Overall, our WBPA approach successfully leveraged the diverse input clusterings in six of seven cases.

8 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. Our ensemble approach WBPA provides a solution to the difficult issue of tuning the input parameter *h* of LAC. In our future work, diversity-accuracy requirements of the individual clusterings, in order for the ensemble to be effective, will be further investigated and quantified.

Acknowledgements Carlotta Domeniconi was in part supported by NSF CAREER Award IIS-0447814. The authors would like to thank Curtis Jamison for helping with the assessment of the biological significance of the generated gene clusters.

References

Aggarwal C, Procopiuc C, Wolf JL, Yu PS, Park JS (1999) Fast algorithms for projected clustering. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 61–72

Aggarwal C, Yu PS (2000) Finding generalized projected clusters in high dimensional spaces. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 70–81

Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 94–105

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

Al-Razgan M, Domeniconi C (2006) Weighted clustering ensembles. In: Proceedings of the SIAM international conference on data mining, pp 258–269

Arabie P, Hubert LJ (1996) An overview of combinatorial data analysis. Clustering and classification. World Scientific, Singapore, pp 5–63

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

Chakrabarti K, Mehrotra S (2000) Local dimensionality reduction: a new approach to indexing high dimensional spaces. In: Proceedings of VLDB, pp 89–100

Cheng Y, Church GM (2000) Biclustering of expression data. In: Proceedings of the 8th international conference on intelligent systems for molecular biology, pp 93–103

Cheeseman P, Stutz J (1996) Bayesian classification (autoclass): theory and results. In: Advances in knowledge discovery and data mining, Chap. 6. AAAI/MIT Press, pp 153–180

Dempster AP, Laird NM, Rubin DB (1997) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc 39(1):1–38

Dhillon IS (2001) Co-clustering documents and words using bipartite spectral graph partitioning. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 269–274

Dhillon IS, Mallela S, Modha DS (2003) Information-theoretic co-clustering. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 89–98

Domeniconi C, Papadopoulos D, Gunopulos D, Ma S (2004) Subspace clustering of high dimensional data. In: Proceedings of the SIAM international conference on data mining, pp 517–520

Duda RO, Hart PE (1973) Pattern classification and scene analysis. Wiley, New York

Dy JG, Brodley CE (2000) Feature subset selection and order identification for unsupervised learning. In: Proceedings of the international conference on machine learning, pp 247–254

Ester M, Kriegel HP, Xu X (1995) A database interface for clustering in large spatial databases. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 94–99

Fern XZ, Brodley CE (2004) Solving cluster ensemble problems by bipartite graph partitioning. In: Proceedings of the international conference on machine learning, pp 281–288

Friedman J, Meulman J (2002) Clustering objects on subsets of attributes. Technical report, Stanford University

Fukunaga K (1990) Introduction to statistical pattern recognition. Academic, New York

Ghahramani Z, Hinton GE (1996) The EM algorithm for mixtures of factor analyzers. Technical report CRG-TR-96-1, Department of Computer Science, University of Toronto

Hartigan JA (1972) Direct clustering of a data matrix. J Am Stat Assoc 67(337):123–129

Hedenfalk I, Duggan D, Chen Y, Radmacher M, Bittner M, Simon R, Meltzer P, Gusterson B, Esteller M, Kallioniemi OP, Wilfond B, Borg A, Trent J (2001) Gene expression profiles in hereditary breast cancer. N Engl J Med 344:539–548

Keogh E, Chakrabarti K, Mehrotra S, Pazzani M (2001) Locally adaptive dimensionality reduction for indexing large time series databases. In: Proceedings of the ACM SIGMOD conference on management of data, pp 151–162

Kharypis G, Kumar V (1995) Multilevel k-way partitioning scheme for irregular graphs. Technical report, Department of Computer Science, University of Minnesota and Army HPC Research Center

Michalski RS, Stepp RE (1983) Learning from observation: conceptual clustering. In: Michalski RS, Carbonell JG, Mitchell TM (eds) Machine learning: an artificial intelligence approach, vol 2. Palo Alto TIOGA Publishing Co., pp 331–363

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

Modha D, Spangler S (2003) Feature weighting in K-means clustering. Mach Learn 52(3):217–237

Ng RT, Han J (1994) Efficient and effective clustering methods for spatial data mining. In: Proceedings of the VLDB conference, pp 144–155

Parsons L, Haque E, Liu H (2004) Subspace clustering for high dimensional data: a review. ACM SIGKDD Explor Newsl 6(1):90–105

Procopiuc CM, Jones M, Agarwal PK, Murali TM (2002) A Monte Carlo algorithm for fast projective clustering. In: Proceedings of the ACM SIGMOD conference on management of data, pp 418–427

Strehl A, Ghosh J (2003) Cluster ensemble—a knowledge reuse framework for combining multiple partitions. J Mach Learn Res 3:583–617

Tipping ME, Bishop CM (1999) Mixtures of principal component analyzers. Neural Comput 1(2):443–482

Thomasian A, Castelli V, Li CS (1998) Clustering and singular value decomposition for approximate indexing in high dimensional spaces. In: Proceedings of CIKM, pp 201–207

Wang H, Wang W, Yang J, Yu PS (2002) Clustering by pattern similarity in large data sets. In: Proceedings of the ACM SIGMOD conference on management of data, pp 394–405

Wu CFJ (1983) On the convergence properties of the EM algorithm. Ann Stat 11(1):95–103

Yang J, Wang W, Wang H, Yu P (2002) δ-Clusters: capturing subspace correlation in a large data set. In: Proceedings of the international conference on data engineering, pp 517–528

Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. In: Proceedings of the ACM SIGMOD conference on management of data, pp 103–114