1 Introduction

Data clustering has a wide range of applications and has been studied extensively in the statistics, data mining and database communities. Many algorithms have been proposed in the area of clustering [1][2]. One popular group of such algorithms, the model-based methods, have sparked wide interest because of their additional advantages, which give them the capacity to describe the underlying structures of populations in the data [3].

In model-based methods, data are thought of as originating from various possible sources, which are typically modeled by Gaussian mixture [3][4][5]. The goal is to identify the generating mixture of Gaussians, that is, the nature of each Gaussian source, with its mean and covariance. Examples include the classical $k$-means [1][2] and its variants. However, such methods would suffer from the curse of dimensionality problem for high-dimensional data [6].

Many types of real-world data, such as the documents represented in the Vector Space Model (VSM) used in text mining and the micro-array gene expression data of bioinformatics, consist of very high-dimensional features. The data are inherently sparse in high-dimensional spaces [7][8], making the Gaussian function inappropriate in this case. Verleysen [7] states that when the dimension increases, the percentage of the samples of a normalized multivariate Gaussian distribution falling around its center would rapidly decrease to 0. In other words, most of the volume of a Gaussian function is contained in the tails instead of near the center in high-dimensional space: the so-called “empty space phenomenon” [7].

Furthermore, in high-dimensional space, clusters may exist in different subspaces comprised of different combinations of features. In many real-world applications, in fact, some points are correlated with a given set of dimensions, and others are correlated with different dimensions. For example, in document clustering, clusters of documents on different topics are characterized by different subsets of keywords. The keywords for one cluster may not occur in the documents of other clusters. To address the above challenges, projective clustering has been defined to find clusters in different subspaces of the same dataset [9][10][11][12][20].

A projected cluster is an ensemble of subsets of points,
each of which is associated with a subset of attributes. In Fig.1, two different projected clusters are illustrated for a set of data points in 3-dimensional space. There are two clusters in this example; however, they are associated with two different low-dimensional subspaces. The first cluster corresponds to the data in group C1, which are close to each other when projected into the subspace consisting of the dimensions A1 and A5, while the second one corresponds to the data in group C2 projected onto the A1–A3 plane.

A number of algorithms for finding such projected clusters have been proposed in the literature. They fall into two categories[12]. Those in the first category, which include PROCLUS[9], ORCLUS[13] and FINDIT[14], are aimed at discovering the exact subspaces of different clusters. The algorithms in the second category cluster data points in the entire data space but assign different weighting values to different dimensions of clusters: examples include EWKM[12], FWKM[15] and LAC[16], most of the algorithms in the second category are of the k-means type, whose sequential structure is analogous to the mathematics of the EM algorithm[17]. However, there is a general lack of underlying models on which these methods can be built.

In this paper, we will present a new model-based method for projective clustering. The first contribution is the proposal of a probability model to describe projected clusters in a high-dimensional space. In contrast to existing models for high-dimensional data clustering, our extended Gaussian model is designed for projective clustering, and by analysis is able to explain the general assumptions used in popular projective methods. Second, we derive an objective function for projective clustering based on the probability model and propose an EM-type, parameter-free algorithm, named MPC, for optimizing the objective function. The performance of MPC has been evaluated on synthetic datasets and some widely used real-world datasets, and the experimental results show its effectiveness. The method presented in this paper is very different from the one in our previous work[13]. Although the basic density function of the projected cluster is reused, the probability model for projected clusters has been changed. This results in a different algorithm which is no more dependent on any user-defined parameter for updating the dimension weights. The new algorithm has been much better motivated, analyzed and experimentally evaluated.

The remainder of this paper is organized as follows. Section 2 presents some related work and the rationale for our work. In Section 3, the projective clustering model is presented. Section 4 describes the new algorithm MPC. Experimental results are presented in Section 5. Finally, Section 6 gives our conclusion and discusses directions for future work.

2 Related Work

2.1 Techniques for High-Dimensional Data Clustering

Techniques for dimensionality reduction have been used in high-dimensional data clustering. Feature transformation techniques, such as PCA and SVD, attempt to summarize the dataset in a smaller number of new dimensions created via linear combination of the original attributes, while feature selection methods select only the most relevant attributes for the clustering task[18]. Because these traditional techniques are performed in the entire data space, they may encounter difficulties when clusters are found in different subspaces. LDR(Local Dimensionality Reduction)[18] attempts to create a new set of dimensions for each cluster. The difficulties with such a method include the determination of dimensionality for each subspace associated with the clusters. Additionally, LDR often has high computational complexity.

Biclustering[21] also referred to as coclustering, has been proposed for simultaneous clustering on the data points and dimensions of high-dimensional data. One of its typical applications is in the analysis of gene expression data, where the task is to find subgroups of genes and subgroups of conditions such that the genes exhibit highly correlated activities for every condition.

Finally, two related terms occur in the literature: subspace clustering and projective clustering. According to Parsons et al.[16], projective clustering algorithms constitute a particular category of the subspace clustering techniques. However, different views are put forward elsewhere in the literature: see for instance[11][23][29]. We adopt the taxonomy of [23] and make a distinction between the two terms based on the ideas behind them. The idea of subspace clustering is to identify all dense regions in all subspaces, whereas in projective clustering the main focus is on discovering clusters that are projected onto particular subspaces. In the subspace clustering field, CLIQUE[19] was the pioneering approach, followed by a number of algorithms such as ENCLUS[19] and MAFIA[16] and SUBCLU[17]. The major concern of this paper is projective clustering. In the following pages, we will focus only on such techniques.

2.2 Projective Clustering Methods

Projective clustering is typically based on feature weighting[19][11]. Each dimension of each cluster is
assigned a weighting value, indicating to what extent the dimension is relevant to the cluster. Usually, the weighting values of a given dimension may be different for different clusters. Based on the way the weights are determined, projective clustering algorithms can be divided into two categories: hard subspace clustering and soft subspace clustering.[12]

In the first category, the dimensions are assigned weights with values of either 0 or 1, resulting in hard feature weighting for the subspaces. PROCLUS[9], which is based on the traditional k-medoids approach, is a representative algorithm using this weighting scheme. PROCLUS samples the data, then selects a set of medoids and iteratively improves the clustering, with the goal of minimizing the average within-cluster dispersion. For each medoid, a set of dimensions is chosen whose average distances to the medoid are small compared to statistical expectation. After the subspaces have been identified, an average Manhattan segmental distance is used to assign points to medoids. PROCLUS requires users to provide the average number of relevant dimensions per cluster, which is usually unknown to users.

FINDIT[14], which uses a distance measure called the Dimension Oriented Distance (DOD), is similar in structure to PROCLUS. As a hierarchical clustering algorithm, HARP[26] automatically determines the relevant attributes of each cluster without requiring user-defined parameters. HARP is based on the assumption that two data points are likely to belong to the same cluster if they are very similar to each other along many dimensions. DOC[27] also defines the subspace as a subset of attributes on which the projection of points in a partition is contained within a segment. DOC computes projected clusters using a randomized algorithm to minimize a certain quality function. MINECLUS[18] improves on DOC by transforming the problem of finding the projected clusters into the problem of mining the frequent itemset.

PROCLUS[9] and the other algorithms mentioned above search for axis-aligned subspaces for the clusters, while some other methods search more general subspaces, termed non-axis-aligned[24], where the new features are linear combinations of the original dimensions. ORCLUS[13] is a generalization of PROCLUS that can discover clusters in arbitrarily oriented subspaces. By covariance matrix diagonalization, ORCLUS selects the eigenvectors corresponding to the smallest eigenvalues of the matrix of the set of points. ORCLUS inherits the weaknesses of PROCLUS mentioned above. KSM[25], a k-means type projective clustering algorithm, determines the non-axis-aligned subspaces by SVD computations, while EPCH[30] performs non-axis-aligned projective clustering by histogram construction.

Instead of identifying hard subspaces for clusters, the algorithms in the second category assign weights in the range [0,1]. Since the weights can be any real number in [0,1], we can call these soft projective clustering algorithms. Typically, the weight value for a dimension in a cluster is inversely proportional to the dispersion of the values from the center in the dimension of the cluster. In other words, a high weight indicates a small dispersion in a dimension of the cluster. Virtually all of the existing algorithms in this category are based on the following general assumptions[25]: (1) the data project along a significant dimension onto a smaller range of values than on the other dimensions; (2) the data are more likely to be uniformly distributed along each irrelevant dimension. We will examine the capabilities of our projective clustering model, presented below, with respect to these two general assumptions.

A number of soft projective clustering algorithms have been reported recently[12][13][10][22][25][36]. In [22], an algorithm makes use of particle swarm optimization is presented. Since a heuristic global search strategy is used, the near-optimal feature weights could be obtained by this algorithm; however, it would run more slowly than other algorithms. To build an efficient soft projective clustering algorithm, the k-means type structure has been widely adopted. Based on the classical k-means clustering process[1], an additional step for computing the weighting values is added in each iteration in these algorithms, which include EWKM[12], FWKM[12], LAC[16], and FSC[25] etc. Algorithm 1 shows a typical structure for these algorithms.

Algorithm 1: A k-mean-type projective clustering algorithm.

From Algorithm 1, the common projective clustering algorithm can be thought of as an EM-based process for estimating the unknown parameters C, V and W of a model F(C,V,W) from which the data originate. However, the underlying F(C,V,W) is generally neglected in the above methods. The lack of such a model makes derivation of more effective clustering algorithms difficult[30]. This has led us to work on projected cluster modeling, since we are convinced this type of modeling process allows us to benefit from the full potential of cluster analysis: for example, in describing the
underlying mechanism that generates the cluster structure and addressing cluster validity problems.

In a typical model-based clustering analysis, one tries to find a mixture of multivariate distributions to approximate the data. Due to the empty space phenomenon and the property of projective clustering, as mentioned above, cluster modeling on high-dimensional data is a difficult problem. In one of the few attempts to use model-based high-dimensional data clustering, Hoff\(^{[19]}\) proposed a model of "clustering shifts in mean and variance" based on a nonparametric mixture of sequences of independent normal random variables. The model is learned by a Markov chain Monte Carlo process; however, its computational cost is prohibitive. Harpaz et al.\(^{[20]}\) presented a non-parametric density estimation modeling technique, where the data are described as a mixture of linear manifolds. The low-dimensional subspaces associated with the individual clusters are computed by PCA. The problems with this method lie in its inflexibility in determining the dimensionality of the subspaces, and its inefficient clustering process.

### 3 A Probability Model for Projective Clustering

The attributes of a non-axis-aligned subspace are typically combinations of the dimensions of the original data space. Since they are difficult to interpret, often making the clustering results less useful for many real applications\(^{[10]}\), such as document clustering, only projected clusters in axis-aligned subspaces are formalized in the following presentation.

#### 3.1 Basic Notation and Definitions

The notation used throughout the paper is summarized in Table 1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_i)</td>
<td>ith data point in (R^D) for (i=1,2,...,N)</td>
</tr>
<tr>
<td>(DB={x_1,x_2,...,x_N})</td>
<td>The data set</td>
</tr>
<tr>
<td>(K)</td>
<td>Number of clusters</td>
</tr>
<tr>
<td>(c_1,c_2,...,c_K)</td>
<td>K clusters of (DB)</td>
</tr>
<tr>
<td>(u_{ik})</td>
<td>Membership degree of (x_i) in (c_k), (k=1,2,...,K)</td>
</tr>
<tr>
<td>(U=(u_{ik})_{K \times N})</td>
<td>Membership matrix, where (k=1,2,...,K) and (i=1,2,...,N)</td>
</tr>
<tr>
<td>(v_{ik})</td>
<td>Cluster center vector of (c_k)</td>
</tr>
<tr>
<td>(P=(v_{ik})_{K \times D})</td>
<td>Cluster center matrix, where (k=1,2,...,K) and (j=1,2,...,D)</td>
</tr>
<tr>
<td>(w_k={w_{k1},w_{k2},...,w_{kD}})</td>
<td>A weight vector associated with (c_k)</td>
</tr>
<tr>
<td>(W=(w_k)_{K \times D})</td>
<td>Weight matrix, where (k=1,2,...,K) and (j=1,2,...,D)</td>
</tr>
</tbody>
</table>

\[0 \leq u_{ik} \leq 1; \sum_{k=1}^{K} u_{ik} = 1, i = 1, 2, ..., N\]  
\[(1)\]

The cluster \(c_k\) is associated with a weight vector\(^3\) \(w_k=\{w_{k1},w_{k2},...,w_{kD}\}\), satisfying

\[\sum_{j=1}^{D} w_{kj} = 1, \quad k = 1, 2, ..., K\]
\[0 \leq w_{kj} \leq 1, \quad k = 1, 2, ..., K; \quad j = 1, 2, ..., D\]  
\[(2)\]

Here the weight \(w_{kj}\) is defined to measure the relevance of the \(j\)th dimension to \(c_k\). The greater the relevance, the larger the weight. Furthermore, we introduce a \(D\times D\) matrix \(s_k\), which is defined as

\[s_k = \begin{bmatrix}
\sqrt{w_{k1}} \\
\sqrt{w_{k2}} \\
... \\
\sqrt{w_{kD}}
\end{bmatrix}\]

For a given \(c_k\) the assignment of \(w_{k1},w_{k2},...,w_{kD}\) can be regarded as a soft feature selection procedure for the space in which \(c_k\) exists\(^{[16]}\). We thus use such a matrix to stand for the subspace associated with a cluster. The definition of projected clusters is given in Definition 1.

**Definition 1.** (Projected Cluster) Let \(PC_k=(s_k,u_k)\) be the \(k\)th projected cluster of \(DB\), where \(u_k=\{u_{k1},u_{k2},...,u_{kN}\}\) denotes the membership degrees of all the data points with regard to the \(k\)th cluster, \(k=1,2,...,K\).

In order to examine the distribution of data points in a projected cluster, we need to project them onto their subspaces.

**Definition 2** (Projection) Let \(x\) be a data point of \(PC_k\) and \(y\) its projection in its subspace,

\[y = \pi_{s_k}(x) = x \cdot s_k\]  
\[(3)\]

Let us examine the above definitions in the following derivation. Considering the Euclidean distance of two projected points \(y_1\) and \(y_2\), we have
The last equation of Eq. (4) is the very weighted Euclidean distance which is commonly used in the existing algorithms [12][15][16][25].

3.2 Probability Model

It is important to note that the Gaussian mixture is a fundamental hypothesis that many model-based clustering algorithms make regarding the data distribution model [39]. In this case, data points are thought of as originating from various possible sources, and the data from each particular source is modeled by a Gaussian. However, Gaussian functions are not appropriate in high-dimensional space due to the curse of dimensionality [7].

In order to learn the underlying structure of clusters in high-dimensional space, we will examine the distribution on each dimension. Consider the projections of the data points of the cluster \( k \) onto the \( j \)th dimension. It is reasonable to describe the projections using a 1D Gaussian function. The probability density function is

\[
G(y_j \mid \mu_y, \sigma_y) = \frac{1}{\sqrt{2\pi\sigma_y}} \exp\left(-\frac{1}{2\sigma_y^2}(y_j - \mu_y)^2\right)
\]

where \( \mu_y \) and \( \sigma_y \) denote the mean and covariance of the Gaussian. Using Definition 2, the density function can be transformed into the original data space. The above expression thus becomes

\[
G(x_j \mid v_y, w_y, \sigma_y) = \frac{1}{\sqrt{2\pi\sigma_y}} \exp\left(-\frac{w_y}{2\sigma_y^2}(x_j - v_y)^2\right)
\] (5)

The major difference between Eq. (5) and the standard Gaussian is the introduction of the weighting value \( w_y \), indicating the contribution of the \( j \)th dimension to \( c_i \). Curves of Eq. (5) are drawn in Fig. 2, with different values of \( w_y \) and a fixed \( \sigma_y \).

By applying the probability model to clustering, the goal is to estimate \( \Theta \) from the given dataset. Supposing \( \hat{\Theta} = (\hat{\alpha}_k, \hat{\beta}_k, \hat{\omega}_k, \hat{\sigma}_k) \mid 1 \leq k \leq K \) is an estimator of \( \Theta \), the distance between \( F(x; \Theta) \) and \( \hat{F}(x; \hat{\Theta}) \) can be measured by the following Kullback-Leibler divergence function [41]:

\[
R(\hat{\Theta}) = \int F(x; \Theta) \ln \frac{F(x; \Theta)}{\hat{F}(x; \hat{\Theta})} \, dx
\]

The equation can be decomposed into two terms. The first, \( \int F(x; \Theta) \ln F(x; \Theta) \, dx \), is a constant that is irrelevant to \( \hat{\Theta} \); therefore, the following objective criterion needs to be maximized (the symbols \( \uparrow \) and \( \downarrow \) in front of the function indicate maximizing and minimizing the function, respectively):
\[ \uparrow Q(\hat{\Theta}) = \int F(x; \Theta) \ln \hat{F}(x; \hat{\Theta}) \text{d}x = \sum_{k=1}^{N} \int p(k \mid x) F(x; \Theta) \ln \hat{F}(x; \hat{\Theta}) \text{d}x \]

with

\[ p(k \mid x) = \frac{\hat{\alpha}_k \prod_{j=1}^{d} \sqrt{\hat{\omega}_j} G(x_j \mid \hat{\nu}_j, \hat{\sigma}_j)}{p(k \mid x)} \tag{7} \]

where \(p(k \mid x)\) is the posterior probability of an input \(x\) from the \(k\)th probability density function, given \(x\). Substituting for \(\hat{F}(x; \hat{\Theta})\) according to Eq. (7) in \(Q(\hat{\Theta})\), we obtain

\[ \uparrow Q(\hat{\Theta}) = \sum_{k=1}^{N} \int p(k \mid x) F(x; \Theta) \ln \frac{\hat{\alpha}_k \prod_{j=1}^{d} \sqrt{\hat{\omega}_j} G(x_j \mid \hat{\nu}_j, \hat{\sigma}_j)}{p(k \mid x)} \text{d}x \tag{8} \]

By the law of large numbers, given a dataset \(DB\), maximizing Eq. (8) is equivalent to the maximum likelihood learning of \(\Theta\) from all the inputs \(x_1, x_2, \ldots, x_N\). Therefore, using Eq. (5) to replace \(G(x_j \mid \hat{\nu}_j, \hat{\sigma}_j)\), the objective criterion can be further transformed into

\[ \downarrow Q_2(\hat{\Theta}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} p(k \mid x_i) \left\{ \frac{1}{2} \sum_{j=1}^{d} \left( \frac{\hat{\omega}_j (x_i - \hat{\nu}_j)^2 - \ln \frac{\hat{\omega}_j}{2 \sqrt{\pi} \sigma_j}}{p(k \mid x_i)} \right) \right\} \]

For an input \(x_i\), the posterior probability \(p(k \mid x_i)\) is thought of as the fuzzy membership \(u_{ki}\) in clustering. Given that \(\hat{\nu}_j\) and \(\sum_{i=1}^{N} \sum_{k=1}^{K} u_{ki} \sum_{j=1}^{d} \ln 2\pi \sigma_j^2\) are constants irrelevant to \(\hat{\Theta}\), the resulting clustering objective function can be obtained as

\[ \downarrow J(U, V, W, Z) = \sum_{i=1}^{N} \sum_{k=1}^{K} \left( \frac{u_{ki}}{2} \sum_{j=1}^{d} \frac{\hat{\omega}_j (x_i - \hat{\nu}_j)^2 - \ln \frac{\hat{\omega}_j}{2 \sqrt{\pi} \sigma_j}}{\alpha_i} \right) \]

subject to the constraints of Eqs. (1), (2) and (6). Here, \(Z = \{ \alpha_1, \alpha_2, \ldots, \alpha_k, \sigma_1, \sigma_2, \ldots, \sigma_k \} \).

4 A Model-Based Algorithm for Projective Clustering

This section presents our algorithm, MPC (Model-based Projective Clustering) for projective clustering by minimizing Eq. (10) subject to the constraints of Eqs. (1), (2) and (6), which is a constrained nonlinear optimization problem. Using the Lagrangian multiplier technique, this can be transformed into an unconstrained optimization problem:

\[ \min J(U, V, W, Z) = J(U, V, W, Z) + \sum_{k=1}^{K} \lambda_k \left( \sum_{i=1}^{N} u_{ki} - 1 \right) \]

\[ + \xi \left( \sum_{i=1}^{N} z_i - 1 \right) + \sum_{i=1}^{N} \zeta_i \left( \sum_{j=1}^{d} w_{ij} - 1 \right) \]

where \(\lambda_k (k=1, 2, \ldots, K)\), \(\xi\) and \(\zeta_i (i=1, 2, \ldots, n)\) are the Lagrange multipliers corresponding to the constraints defined in Eqs. (1), (2) and (6).

4.1 The Optimization Method

To achieve a local minimum of the objective function, the usual method is to use the partial optimization for each parameter in the function\(^{13,12,15}\). Following this method, minimization of \(J\) in Eq. (11) can be performed by optimizing \(U, V, W\) and \(Z\) in a sequential structure analogous to the mathematics of the EM algorithm\(^{17}\). In each iteration, we first fix \(V = \hat{V}, W = \hat{W}\) and \(Z = \hat{Z}\), and solve \(U\) as \(\hat{U}\) to minimize \(J(U, \hat{V}, \hat{W}, \hat{Z})\). Next, we fix \(U = \hat{U}\), \(W = \hat{W}\) and \(Z = \hat{Z}\) and solve \(V\) as \(\hat{V}\) to minimize \(J(\hat{U}, \hat{V}, \hat{W}, \hat{Z})\). Then, \(U = \hat{U}\), \(V = \hat{V}\) and \(W = \hat{W}\) are fixed and the optimal \(Z\), say \(\hat{Z}\), is solved to minimize \(J(U, \hat{V}, \hat{W}, \hat{Z})\). The four partial optimization problems are solved according to the following theorems.

**Theorem 1.** Let \(V = \hat{V}, W = \hat{W}, \text{ and } Z = \hat{Z}\) be fixed, \(J(U, \hat{V}, \hat{W}, \hat{Z})\) is minimized iff

\[ u_k = \frac{\bar{u}_k}{\sum_{i=1}^{N} \bar{u}_i} \tag{12} \]

with

\[ \bar{u}_k = \hat{\alpha}_k \prod_{j=1}^{d} \frac{\hat{\omega}_j}{\hat{\sigma}_j} \exp \left( - \frac{1}{2 \hat{\sigma}_j^2} \sum_{j=1}^{d} \hat{\omega}_j (x_i - \hat{\nu}_j)^2 \right) \]

**Proof.** The proof is given in Appendix A.

**Theorem 2.** Let \(U = \hat{U}, W = \hat{W}\) and \(Z = \hat{Z}\) be fixed, \(J(\hat{U}, V, \hat{W}, \hat{Z})\) is minimized iff

\[ \hat{v}_k = \frac{1}{\sum_{i=1}^{N} \bar{u}_i} \sum_{i=1}^{N} \bar{u}_i x_i \]

**Proof.** The proof is given in Appendix B.

**Theorem 3.** Let \(U = \hat{U}, V = \hat{V}\) and \(W = \hat{W}\) be fixed, \(J(\hat{U}, \hat{V}, \hat{W}, Z)\) is minimized iff
\[ \hat{\sigma}_k = \frac{\sum_{i=1}^{N} \hat{u}_{ki} X_{iy}}{N} \quad (14) \]

and

\[ \hat{\sigma}_k^2 = \frac{1}{D} \sum_{i=1}^{D} \sum_{j=1}^{N} \hat{u}_{ki} X_{ij} \quad (15) \]

with

\[ X_{ij} = \frac{\hat{u}_{ki} (x_i - \hat{v}_k)^2}{1+2\hat{\sigma}_k^2 \lambda_k} \]

denoting the dispersion of the data points from the center in the dimension of the cluster.

**Proof.** The proof is given in Appendix C.

**Theorem 4.** Let \( U = \hat{U}, \hat{V} = \hat{V} \) and \( Z = \hat{Z} \) be fixed, \( J(\hat{U}, \hat{V}, \hat{W}, \hat{Z}) \) is minimized iff

\[ \hat{w}_{ij} = \frac{\hat{\sigma}_k^2 \sum_{i=1}^{N} \hat{u}_{ki} X_{ij}}{X_{ij} + 2\hat{\sigma}_k^2 \lambda_k} \quad (16) \]

with

\[ \sum_{j=1}^{D} X_{ij} + 2\hat{\sigma}_k^2 \lambda_k = \frac{1}{\hat{\sigma}_k^2 \sum_{i=1}^{N} \hat{u}_{ki}} \quad (17) \]

**Proof.** The proof is given in Appendix D.

Here are some comments on the above equations for computing a local minimal solution of the problem in Eq. (11). By Eq. (13), we can compute the optimal center of a projected cluster, which is the same as the one defined in FCM \cite{FCM} (in the case where the fuzzifier is equal to 1). The value of \( \hat{\sigma}_k^2 \) can be thought of as the “compactness” of the projected cluster \( PC_k \). Furthermore, the \( 2\hat{\sigma}_k^2 \lambda_k \) in Eq. (16) prevents the denominator from being zero (in the next subsection we will show \( X_{ij} + 2\hat{\sigma}_k^2 \lambda_k \) always lies within a particular range), making the feature weights computable in practice.

### 4.2 The MPC Algorithm

The MPC algorithm, as outlined by Algorithm 2, performs projective clustering by minimizing the objective function of Eq. (11), using the optimization methods presented in the previous subsection. Actually, this solution can also be regarded as an extension to the classical FCM algorithm \cite{FCM} by adding an additional step in each iteration to compute \( W \) for each cluster, an approach which is commonly adopted in existing soft subspace clustering algorithms such as \cite{PROCLUS,FWKM,EWKM,FSC}.

It is important to note that MPC does not require user-defined parameters for feature weighting, whereas most of the existing projective clustering algorithms do: for instance, \( n \) in PROCLUS \cite{PROCLUS}, \( \beta \) in FWKM \cite{FWKM}, \( \gamma \) in EWKM \cite{EWKM}, etc. The only pending coefficient, say \( \hat{\lambda}_k \), in the weight updating formula (Eq. (16)) of MPC can be determined by numerically solving Eq. (17). Step (4) of Algorithm 2 is designed for this purpose. In Eq. (17), all the variables except \( \hat{\lambda}_k \) are given and thus can be considered as constants with respect to \( \hat{\lambda}_k \), consequently, we can resolve \( \hat{\lambda}_k \) from Eq. (17) using a numerical method, such as the Newton-Raphson and bisection method \cite{Numerical_Methods}. Theorem 5 states that \( \hat{\lambda}_k \) always lies within a particular range.

**Algorithm 2:** The outline of the MPC algorithm.

**Theorem 5.** The root \( \hat{\lambda}_k \) of Eq. (17) is unique in \((-\infty, +\infty)\), where \( \text{MIN}_{X_k} \) denotes the minimum of \( X_{ij} \) for \( j=1,2,...,D \).

**Proof.** In the following, the equation

\[ f(\hat{\lambda}_k) = \sum_{j=1}^{D} \frac{X_{ij}^2}{X_{ij} + \hat{\lambda}_k} \]

is considered, where

\[ \hat{\lambda}_k = 2\hat{\sigma}_k^2 \hat{\lambda}_k \text{. Because } \frac{\partial f(\hat{\lambda}_k)}{\partial \hat{\lambda}_k} < 0, \text{ monotonically decreasing while } \hat{\lambda}_k \text{ falls in } (-\infty, +\infty). \]

On the other hand, \( \lim_{\hat{\lambda}_k \to +\infty} f(\hat{\lambda}_k) > 0 \) and \( \lim_{\hat{\lambda}_k \to -\infty} f(\hat{\lambda}_k) < 0 \). Theorem 5 is obtained.

Theorem 5 also implies that \( X_{ij} + 2\hat{\sigma}_k^2 \hat{\lambda}_k > 0 \) for \( j=1,2,...,D \). This consequence indicates that the denominator of our weighting formula in Eq. (16) is always greater than zero. By comparison, in the weighting formula of FWKM \cite{FWKM} and FSC \cite{FSC}, a user-defined constant has to be added to make the denominator greater than zero.
The computational complexity of MPC is $O(PKND)$, where $P$ denotes the number of iterations. Since the Newton-Raphson and bisection method, which is an iterative algorithm, is used for Step (4), it could increase the running time of MPC in practice. The scalability will be examined in the next section. Below, we will show that MPC is able to converge in a finite number of iterations. First, it can be seen that the objective function is bounded below, according to Theorem 6.

**Theorem 6.** Let $p(p>0)$ be an iteration index of MPC, $J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)}) > N(1 - K)$. 

**Proof.** The proof is given in Appendix E.

Next, we show that the values of the objective function are strictly decreasing over the iterative processes of MPC. Let $p$ be an iteration index and $p > 0$. In Step .1 of the $(p+1)$th iteration, the objective function is partially minimized by updating $U^{(p)}$ to $U^{(p+1)}$, according to Theorem 1. Thus, $J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)}) \leq J(U^{(p)}, V^{(p)}, W(p), Z(p))$ at the end of Step (1). By applying Theorem 2 and 3, we obtain $J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)}) \leq J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)})$ and $J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)}) \leq J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)})$. The objective function is finally optimized via steps (4) and (5), according to Theorem 4, so that $J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)}) \leq J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)})$. In general, we have $J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)}) \leq J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)})$. In that case, the algorithm would have stopped since the value of the objective function remains the same. Thus, MPC always reduces the objective function $J(\cdot, \cdot, \cdot, \cdot)$, which is also bounded below. Given $\varepsilon > 0$, the MPC algorithm converges in a finite number of iterations. Obviously, the number of iterations is dependent on the dataset and on the initial selection of $V$ and $\varepsilon$.

## 5 Experimental Evaluation

In this section, we evaluate the performance of MPC on synthetic and real-world datasets, and we also experimentally compare MPC with a few other mainstream clustering algorithms. All experiments were conducted on a PC with Intel Duo CPU of 2.4GHz and 2GB RAM.

### 5.1 Experimental Setup and Evaluation Measures

Five projective clustering algorithms, including PROCLUS[9], EWKM[12], LAC[16], FSC[33] and FWKM[44], have been chosen as the competing algorithms in the experiments. We chose PROCLUS[9] for the performance comparison because it is the benchmark projective clustering algorithm in the literature. EWKM[22], LAC[16], FSC[33] and FWKM[44] are recently published algorithms that are known as soft subspace clustering algorithms in the literature, and have similar algorithm structures to MPC. In addition, we also compare the performance with two mainstream fuzzy clustering algorithms, the classical fuzzy algorithm FCM[41] and the algorithm Fuzzy-FWKM(Fuzzy Feature Weighting k-Means)[43].

While MPC does not need additional user-defined parameters other than the number of clusters $K$, each of the competing algorithms does. In particular, the user must specify the fuzzifier $m$ in advance to use FCM[41], the fuzzifier $\alpha$ and the weighting exponent $\beta$ in Fuzzy-FWKM[43], and the average number of relevant dimensions per cluster $l$ in PROCLUS[9], FWKM[34], FSC[12], LAC[16] and FSC[33] require their own parameters $\beta$, $\gamma$, $h$ and $\alpha$, respectively, to control the strength of the incentive for clustering on more dimensions. In our experiments, we have used the most common setting, or author-recommended values, for these parameters. For instance, we have used $m=2.0$ in FCM, $\alpha=2.0$ and $\beta=2.0$ in Fuzzy-FWKM, $\beta=1.5$ in FWKM, $\gamma=0.5$ in EWKM, $h=1.9$ in LAC, $\alpha=2.1$ in FSC, and different values for $l$ in PROCLUS[9] according to the characteristics of the datasets.

To measure the quality of the clustering results, we have adopted two main approaches. One is based on the use of a cluster validity index(CVI)[40] which evaluates the performance independent of point labeling. In our experiments, we have employed the well-known partition coefficient $V_{pc}$, which is defined as follows[44]:

$$V_{pc}(U) = \frac{1}{N} \sum_{i=1}^{K} \sum_{u \in U_i} \mu_u^2$$

Here, the value of $V_{pc}$ depends on the fuzziness of the memberships, which is directly related to overlap between pairs of clusters. The larger the values of $V_{pc}$, the less fuzzy the memberships. In this case, the algorithm is considered to be more capable of overcoming overlap between clusters. Such a measure, based on fuzzy membership, cannot be used to judge crisp clustering algorithms, which is why it is not computed for the algorithms FWKM, EWKM, LAC, FSC and PROCLUS below.

The other approach for evaluating the performance of a clustering is a classification-based one, where each cluster is associated with a "class" label corresponding to the (relative) majority class. This approach requires that the ground truth of the datasets be known, which is the case in our experiments. We first transform the fuzzy assignments into hard ones (defuzzification) by identifying the cluster with the highest degree of
membership for each data point. Formally, each data point \( x_i \) is considered to belong to cluster \( c_l \) determined by the following rule:

\[
l = \arg \max_{a=1\ldots K} u_{il}
\]

Then each cluster is labeled by the majority label of its members. Consequently, the conventional measures of recall, precision and F-measure can be used. In particular, we have made use of the FScore, which is defined as follows\(^{12}\):

\[
FScore = \frac{\sum_{l=1}^{K} N_l \max_{j \in \mathcal{S}} \left\{ \frac{2 \times R(\text{class}_k, c_l) \times P(\text{class}_k, c_l)}{R(\text{class}_k, c_l) + P(\text{class}_k, c_l)} \right\}}{N}
\]

where \( R(\text{class}_k, c_l) \) and \( P(\text{class}_k, c_l) \) are the recall and precision values of the \( k \)th class \( \text{class}_k \) with respect to the \( l \)th cluster \( c_l \), and \( N_l \) is the size of \( c_l \). \( R(\text{class}_k, c_l) \) is defined as \( N_k \cap N_{cl} \), and \( P(\text{class}_k, c_l) \) is defined as \( N_k / N_l \), where \( N_l \) is the size of \( c_l \) and \( N_k \) the number of data points occurring in both class \( \text{class}_k \) and cluster \( c_l \).

### 5.2 Experiments on Synthetic Datasets

This first set of experiments was conducted on synthetic datasets. The use of synthetic datasets is motivated by the fact that their cluster structures (syonym here to the difficulty for clustering) can be controlled and their ground truth is known, which facilitates performance evaluation.

#### 5.2.1 Synthetic Data Generation Method

The synthetic datasets were generated using a method similar to that suggested by Aggarwal et al.\(^{9}\). The datasets are characterized by the parameters used in the data generation process, including \( N \) (the number of data points), \( D \) (the dataset dimensionality), \( K \) (the number of clusters), \( l \) (the average number of relevant dimensions per cluster), and \( r \) (the spread parameter) and \( s \) (the scale factor) to control the variance of each cluster on the relevant dimensions.

The major characteristics of the generated datasets were controlled as follows. The dimensions for each cluster were generated using an iterative technique\(^{9}\) to model the fact that different clusters share subsets of correlated dimensions. On an irrelevant dimension of a given cluster, the coordinates of the points are uniformly distributed in the range \([0,100]\). On the other hand, the coordinates of the points on a relevant dimension are generated according to a normal distribution, with the mean randomly chosen from \([0,100]\) and the variance on the \( j \)th dimension determined by \((s_j \times r)^2\), where \( s_j \) is chosen uniformly at random in the range \([1,s]\). For our data generation, \( r=2 \) and \( s \) was varied for different datasets to simulate certain phenomena of cluster overlapping.

To evaluate the capability of MPC to correctly identify projected clusters in various subspaces, two groups, each containing seven datasets, were generated with \( N=5000 \), \( K=5 \). For the first group, \( D=60 \), \( l=12 \) and the scale factor \( s \) varies from 2 to 8 with increment 1 for each dataset. The aim of this group is to simulate a variety of cluster overlaps which make the datasets increasingly difficult to cluster. Table 2 shows a fragment of the dataset with \( s=8 \), where the elements represent the mean \( (\mu) \) and the standard deviation \( (\sigma) \) of clusters on some dimensions. The dimensions with values in bold typeface are considered as the relevant ones to the cluster. Note that the standard deviation on an irrelevant dimension is \( \frac{100}{25} \times 28.87 \) on average. In this case, it is difficult to distinguish the cluster boundary since the data points belonging to different clusters are usually overlapped on the relevant dimensions that they share. For instance, on the dimension 12, most of the data points of \( c_2 \) fall in the range\(^{6} \[(48.88,97.60) \]. The range corresponding to \( c_2 \) is \([-5.05,72.67] \). It can be seen that there are significant overlaps between the two clusters on the dimension.

### Table 2. Distributions of the Dataset with $s$s=8 on Some Dimensions

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Dimension 13</th>
<th>Dimension 14</th>
<th>Dimension 37</th>
<th>Dimension 42</th>
<th>Dimension 46</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>72.95</td>
<td>12.18</td>
<td>50.51</td>
<td>29.28</td>
<td>49.39</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>31.86</td>
<td>19.63</td>
<td>50.59</td>
<td>29.37</td>
<td>51.61</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>48.98</td>
<td>28.90</td>
<td>38.80</td>
<td>28.46</td>
<td>49.38</td>
</tr>
<tr>
<td>( c_4 )</td>
<td>16.95</td>
<td>27.81</td>
<td>46.90</td>
<td>4.38</td>
<td>67.27</td>
</tr>
<tr>
<td>( c_5 )</td>
<td>53.01</td>
<td>27.66</td>
<td>47.69</td>
<td>28.63</td>
<td>45.74</td>
</tr>
</tbody>
</table>

For the second group, we set \( D=100 \), \( s=6 \) and let the average cluster dimensionality \( l \) vary from 5% to 40% of the dimensionality \( D \) for each dataset. The purpose of this group is to provide challenges in terms of the number of relevant dimensions. As shown in [22], clusters with a very small number of relevant dimensions compared to the dimensionality of the data space usually pose the greatest challenges to projective clustering algorithms. Finally, to evaluate the scalability of MPC, we also generated datasets by varying either the size \( N \), the number of dimensions \( D \), or the number of clusters \( K \).

#### 5.2.2 Evaluation Results

Each dataset from the first group was clustered by each algorithm for 20 executions and both the best and average performances are reported. This is because all eight algorithms choose their initial cluster centers via some random selection methods, and thus the clustering results may vary depending on the initialization. Figures

\(^{6}\) According to the properties of a normal distribution, about 95% of the values are within two standard deviations. The lower bound of the range can be computed as 73.24-2×12.18=48.88, and the upper bound is 73.24+2×12.18=97.60.
3 and 4 show the average results of the algorithms on these datasets, in terms of \( V_{PC} \) and FScore respectively. Detailed clustering results on the dataset with \( s=8 \), which is the most difficult case of the seven datasets (as shown in Table 2), are illustrated in Table 3. The values in the max columns correspond to the best results of the algorithms, and the average results are reported in the format average ± 1 standard deviation in the table.

Figures 3 and 4 show that MPC is able to achieve high-quality overall results, especially when the clusters overlap considerably, whereas FCM, Fuzzy-FWKM and EWKM perform poorly, and the other algorithms encounter difficulties when the cluster overlapping becomes significant, i.e., when \( s>6 \). Examining these results in more detail, we can see that the values of \( V_{PC} \) yielded by FCM and Fuzzy-FWKM are close to \( 1/K \), which indicates that these two algorithms tend to assign each point to all the clusters with approximately equal membership degrees. This is due to the fact that FCM measures the similarity between data points by considering all features of a dataset. With the high-dimensional data used in the experiments, such a similarity measurement in the entire data space would be less meaningful due to the empty space phenomenon[8]. Fuzzy-FWKM employs a feature weighting mechanism in the clustering process; however, each dimension is assigned the same weight for different clusters in this algorithm. This is not effective because Fuzzy-FWKM tries to discover the clusters in the same projected subspace.

The performances of FWKM, LAC and FSC are comparable to that of MPC when \( s<6 \). On the other hand, their performances drop rapidly when \( s>6 \). The experiments show that these crisp clustering algorithms encounter difficulties in identifying projected clusters that are highly overlapping on some relevant dimensions. EWKM performs poorly compared with the other five projective clustering algorithms even when \( s<6 \). Although it is always possible that the weak performances of these competing algorithms are due to non-optimal choices of different parameter values, there are some more fundamental reasons. In fact, much like the k-means/FCM, these four algorithms make use of distance-based functions to compute the memberships. Distance-based approaches are known to be ineffective in dealing with cluster overlap, because the variance of a cluster on each dimension is not taken into consideration.

The experiments also show that PROCLUS\(^5\) is very sensitive to the value of \( s \). PROCLUS determines the subspace for each cluster by a hard feature selection method[9]. An underlying relevant dimension will not be selected if the variance of the cluster on that dimension is not small enough, which is often the case for an overlapped cluster.

<table>
<thead>
<tr>
<th></th>
<th>( V_{PC} ) max</th>
<th>avg ± std</th>
<th>FScore max</th>
<th>avg ± std</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPC</td>
<td>0.91</td>
<td>0.85 ± 0.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fuzzy-FWKM</td>
<td>0.20</td>
<td>0.54 ± 0.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCM</td>
<td>0.20</td>
<td>0.66 ± 0.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EWKM</td>
<td>-</td>
<td>0.69 ± 0.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAC</td>
<td>-</td>
<td>0.39 ± 0.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSC</td>
<td>-</td>
<td>0.31 ± 0.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROCLUS</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^5\) The parameter \( l \) of PROCLUS was set to 12 in this set of experiments, which is the exact value we used to generate the data.
variance can be written as $\sigma^2_{j/k}$. On the other hand, the fact that each of these variances is factored into a cluster-dependent variance $\sigma^2_j$ and a dimension-dependent weight parameter $w_j$ improves the efficiency of the parameter optimization process. In fact, $\sigma^2_j$ corresponds to the “size” of cluster $k$ and $w_j$, for $j=1,...,D$, correspond to the “shape” of each cluster. Thus there is a clear geometric interpretation for the dimension weights in MPC and their optimization is more closely linked to the distribution of the data. This is not the case for the four competing soft projective clustering algorithms. For instance, in EWKM, the weights are given a probabilistic interpretation. They are optimized as contributing factors to an (entropy-based) penalty function whose inclusion in the general objective function is only based on a heuristic idea of stimulating dimensions to contribute to the identification (detection) of clusters.

The second set of experiments, designed to analyze the impact of cluster dimensionality on clustering quality, was conducted on the datasets of the second group. Again, each dataset was clustered by each algorithm for 20 executions, and the average performances are reported. Fig. 5 shows the average FScore of the algorithms on these datasets. As we can see from Fig. 5, MPC can achieve accurate clustering results with varying cluster dimensionality. In the case of $1 < 10\%$ of $D$, MPC also yields high-quality results with more than 15% improvement in terms of FScore over the other algorithms. The experiments show that MPC is more robust to the variation of cluster dimensionality than other mainstream projective clustering algorithms.

All of the competing algorithms perform poorly when the average cluster dimensionality is lower than 10% of $D$. The dimension selection mechanism of PROCLUS is based on a distance calculation that involves all dimensions, which may discard relevant dimensions when the cluster dimensionality decreases\[22\]. FWKM and FSC utilize a similar mechanism for dimension selection, in which a constant has to be added to the distance function in order to make the feature weights computable. The constant used in FWKM is computed based on the average dispersion of the entire dataset for all the dimensions\[34\], which may be less valid when the number of relevant dimensions is much lower than that of the entire space. FSC instead uses a small constant independent of the dataset\[33\], resulting in its low suitability in such cases. Both LAC and EWKM use an entropy-based method for dimension selection, the concern being to stimulate the use of more dimensions. The distribution of feature weights may thus be severely biased when most of the dimensions are irrelevant to the clusters. As discussed previously, in the model-based approach of MPC, weights correspond to cluster shapes. Irrelevant dimensions result in very large dimension-dependent variances $\sigma^2_{j/k}$. Due to the dimension-independent cluster size parameter $\sigma^2_j$, determined by a trade-off of all the dimensions, the dimension weights $w_j$ corresponding to large $\sigma^2_{j/k}$ are forced to become very small, which, in turn, reduces the effect of these irrelevant dimensions in cluster formation - for instance, in the membership computation (see Eq. (12)).

5.2.3 Scalability

The scalability of MPC with respect to the data size and the numbers of dimensions and clusters is tested in this subsection. Fig.6 shows the average times in seconds
used by MPC, as well as the average FScore returned by MPC, on the datasets in various situations. All the datasets used here were generated using the same method as discussed in the previous subsection.

Fig. 6(a) illustrates the relationships between the runtime and the data size. The datasets in this experiment contain 4 clusters, each existing in some 8-dimensional subspace of a 20-dimensional data space. From the figure we can see that the runtime of MPC increases linearly with respect to the number of data points, while the clustering accuracies are close to 1 for all the datasets. The scalability with respect to the number of dimensions was tested on synthetic datasets containing 20,000 data points and 4 clusters. The average number of relevant dimensions of the clusters is 40% of the dimensionality of the entire space. The sensitivity with respect to the dimensionality of the space is illustrated in Fig. 6(b), from which we can see that MPC scales linearly with the dimensionality, accompanied by high clustering quality.

Finally, we generated synthetic datasets containing 20,000 data points in 20-dimensional space to test the scalability with respect to the number of clusters. Each cluster exists in an 8-dimensional subspace. Fig. 6(c) shows that the runtime of MPC scales sub-linearly with respect to the number of clusters. The FScore is affected by the number of clusters, since all the data are generated in a limited range [0,100], while the number of clusters is increasing. In this case, the greater the number of clusters, the larger the possibility that clusters are overlapped. Consequently, the clustering accuracy is affected.

5.3 Experiments on Real-world Datasets

Below, we present the experimental results of the MPC algorithm and other projective clustering algorithms on some widely used real-world datasets.

5.3.1 Real-world Datasets

Five datasets were used in this set of experiments. We obtained two datasets, namely Wisconsin Breast Cancer (WBC) and Wine, from the UCI Machine Learning Repository (ftp.ics.uci.edu/pub/machine-learning-databases). The WBC dataset contains 699 instances on clinical cases, where each instance belongs to one of two possible classes: benign or malignant. Each instance is presented with 9 features in the dataset. The Wine dataset contains quantities of 13 constituents extracted from the results of a chemical analysis of wines derived from three different cultivars, which are considered as the three classes of the dataset. Table 4 lists the details of the datasets.

The other three datasets were extracted from popular corpora containing email documents. The data are typically used for spam filtering, which is generally applied in grouping email collections into two categories corresponding to non-spam and spam. The first dataset Email1431 contains 789 spam and 642 non-spam emails, and has been referred to in related work.[10][46] The second one is Ling-Spam,[46], consisting of 481 spam and 2412 non-spam emails. Enron-Spam[47], the third email dataset used in our experiments, is a relatively large collection of documents, which contains 1459 spam and 3669 non-spam documents. Both Ling-Spam and Enron-Spam are available at http://www.iit.demokritos.gr/skel/i-config/.

The email documents are represented in a Vector Space Model(VSM), where each document is a vector in the word space and each element of the vector indicates the frequency of the corresponding word in the document.

For each corpora, we first used the preprocessing method described in [34] to compute the VSM model, and removed stopwords using the common stoplists. Then, the 1000 most frequent words were selected for the final dataset. Each vector $x$ of the dataset was scaled to unit length such that $||x||_2=1$. Finally, all the datasets were normalized using the min-max normalization method.[31]

5.3.2 Experimental Results

Six clustering algorithms, MPC, PROCLUS[31], EKWM[42], LAC[46], FSC[33] and FWKM[34] were tested on the real-world datasets. Since FCM[41] and Fuzzy-FWKM[43] are not projective clustering algorithms, we left them out of this set of experiments. The performances of the algorithms were measured in terms of FScore.

Table 5 illustrates the clustering results returned by each algorithm on its 20 clustering processes. The two figures in each cell represent the maximal and average FScore values. The latter is in the format average ± 1 standard deviation. The best clustering results are marked in bold face. From the table we can see that MPC is able to

<p>| Table 4. Details of the Real-world Datasets |</p>
<table>
<thead>
<tr>
<th>Datasets</th>
<th>Dimensions</th>
<th>Classes</th>
<th>Data Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>WBC</td>
<td>9</td>
<td>2</td>
<td>699</td>
</tr>
<tr>
<td>Wine</td>
<td>13</td>
<td>3</td>
<td>178</td>
</tr>
<tr>
<td>Email1431</td>
<td>1000</td>
<td>2</td>
<td>1431</td>
</tr>
<tr>
<td>LingSpam</td>
<td>1000</td>
<td>2</td>
<td>1893</td>
</tr>
<tr>
<td>EnronSpam</td>
<td>1000</td>
<td>2</td>
<td>5128</td>
</tr>
</tbody>
</table>

<p>| Table 5. Comparison of clustering results on the real-world datasets |</p>
<table>
<thead>
<tr>
<th>Datasets</th>
<th>MPC</th>
<th>FWKM</th>
<th>EKWM</th>
<th>LAC</th>
<th>FSC</th>
<th>PROCLUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>WBC</td>
<td>0.95</td>
<td>0.89</td>
<td>0.89</td>
<td>0.96</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>0.95±0.00</td>
<td>0.89±0.00</td>
<td>0.72±0.05</td>
<td>0.96±0.00</td>
<td>0.94±0.00</td>
<td>0.75±0.04</td>
</tr>
<tr>
<td>Wine</td>
<td>0.97</td>
<td>0.97</td>
<td>0.90</td>
<td>0.95</td>
<td>0.94</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>0.97±0.02</td>
<td>0.97±0.01</td>
<td>0.79±0.05</td>
<td>0.95±0.00</td>
<td>0.94±0.00</td>
<td>0.75±0.04</td>
</tr>
<tr>
<td>Email1431</td>
<td>0.99</td>
<td>0.98</td>
<td>0.97</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>0.99±0.02</td>
<td>0.98±0.01</td>
<td>0.94±0.03</td>
<td>0.98±0.01</td>
<td>0.98±0.01</td>
<td>0.98±0.01</td>
</tr>
<tr>
<td>LingSpam</td>
<td>0.98</td>
<td>0.97</td>
<td>0.96</td>
<td>0.98</td>
<td>0.96</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td>0.98±0.01</td>
<td>0.96±0.01</td>
<td>0.94±0.03</td>
<td>0.98±0.01</td>
<td>0.98±0.01</td>
<td>0.98±0.01</td>
</tr>
<tr>
<td>EnronSpam</td>
<td>0.80</td>
<td>0.87</td>
<td>0.79</td>
<td>0.70</td>
<td>0.84</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>0.80±0.00</td>
<td>0.79±0.00</td>
<td>0.70±0.04</td>
<td>0.72±0.00</td>
<td>0.71±0.00</td>
<td>0.67±0.04</td>
</tr>
</tbody>
</table>
achieve high-quality results on all the datasets. On the two UCI datasets, which have relatively low dimensionality and small data size, FWKM, EWKM and FSC obtained high clustering quality comparable to that of MPC and LAC. The performances of PROCLUS were affected by its parameter setting. The average cluster dimensionality parameter of PROCLUS is difficult to estimate on real-world datasets, we set the parameter to $D/3$ in this set of experiments, where $D$ is the dimensionality of the datasets. For the three email datasets, MPC achieved outstanding performances in terms of the average FScore compared with the other algorithms. We also observe that FWKM, EWKM, LAC and FSC yielded results of competitive quality in terms of the maximal FScore; however, from the standard deviations, we can see that these algorithms are sensitive to the initialization. For instance, on the LingSpam dataset, the maximal FScore of LAC was only 0.88 and the standard deviation reached 0.10. In general, MPC is more robust than the other algorithms. This can be explained by the observation of Jain et al.\cite{1] that fuzzy clustering is usually better than hard clustering at avoiding local minima.

The experiments show that MPC is suitable for clustering real-world data, especially for email documents. To confirm the suitability of our algorithm for document clustering, the capability of MPC in identifying the keywords of document categories is analyzed below. From the subspaces of resulting clusters, we can obtain the relevant dimensions that represent important keywords by sorting the dimension weights in descending order. Table 6 gives some examples for the three email datasets used in the experiments. From the table, we can see MPC has identified important keywords that indicate the spam category in the email collections, such as cash and sex of Email1431, advertisement and unsolicit for LingSpam, and play and hot for EnronSpam. In addition, MPC is parameter-free, making it more practical than other algorithms in real-world applications such as document clustering.

## 6 Conclusion and Perspectives

In this paper, we first discussed the problem of providing a probability model to describe projected clusters in high-dimensional data. This problem becomes difficult due to the sparsity of high-dimensional data and the fact that only a small number of the dimensions may be considered in the clustering process. We proposed an extended Gaussian model which meets the general requirements of projective clustering well. We also derived an objective clustering criterion based on the model, allowing the use of a k-means-type paradigm. By mathematical derivations, we obtained computational expressions for calculating the optimal values of the parameters automatically, and proposed a fuzzy clustering algorithm named MPC. The experiments were conducted on synthetic datasets, UCI datasets and email corpora widely used in real-world applications, and the results show the effectiveness of MPC.

There are many directions that are clearly of interest for future exploration. One avenue of further study is to extend MPC to the case of non-axis-aligned subspaces. Another interesting extension would be for the detection of possible outliers in a dataset. Our future efforts will also be directed toward developing techniques to build a robust initial condition for the clustering algorithm.

### Appendix A Proof of Theorem 1

**Proof.** When $Y = \hat{Y}$, $W = \hat{W}$ and $Z = \hat{Z}$ are fixed, the problem of Eq. (11) can be decomposed into $N$ independent minimization problems:

$$\min_{\phi_k(u_i', \zeta_i)} = \frac{1}{2} \sum_{i=1}^{K} \left( \frac{\hat{w}_i(x_y - \hat{v}_i)^2}{\hat{\sigma}_i^2} - \ln \frac{\hat{w}_i}{u_i} \right)$$

where $u_i', \zeta_i$ minimizes $\phi_k(u_i', \zeta_i)$, its gradients with respect to the variables should vanish, i.e., $\frac{\partial \phi_k(u_i', \zeta_i)}{\partial u_i} = 0$ for $k = 1, 2, \ldots, K$ and $\frac{\partial \phi_k(u_i', \zeta_i)}{\partial \zeta_i} = 0$. Thus we obtain

$$-\frac{1}{2} \sum_{i=1}^{K} \frac{\hat{w}_i}{\hat{\sigma}_i^2} (x_y - \hat{v}_i)^2 - \ln \frac{\hat{w}_i}{u_i} \frac{\hat{\sigma}_i}{u_i} = 1 + \hat{\zeta}_i = 0$$

and $\sum_{i=1}^{K} \hat{u}_i = 1$.
Solving the two equations, the result of Theorem 1 (Eq. (12)) follows.

Appendix B Proof of Theorem 2

Proof. When \( U=\tilde{U} \), \( W=\tilde{W} \) and \( Z=\tilde{Z} \) are fixed, we consider the following \( K \times D \) independent minimization problems from the problem of Eq. (11):

\[
\min \phi_y(v_y) = \sum_{j=1}^{D} \hat{u}_y(x_y - v_y)^2
\]

where \( k=1,2,\ldots,K \) and \( j=1,2,\ldots,D \). Let \( \hat{v}_y \) be the value that minimizes \( \phi_y(v_y) \). Setting the gradient to zero yields

\[
\frac{\partial \phi_y(\hat{v}_y)}{\partial \hat{v}_y} = -2 \sum_{j=1}^{D} \hat{u}_y (x_y - v_y) = 0
\]

Therefore, the result of Theorem 2 follows.

Appendix C Proof of Theorem 3

Proof. When \( U=\tilde{U} \), \( V=\tilde{V} \) and \( W=\tilde{W} \) are fixed, the following minimization problem can be defined based on Eq. (11):

\[
\min \phi_z(Z, \xi) = \xi \left( \sum_{k=1}^{K} \alpha_{k} - 1 \right) + \sum_{k=1}^{K} \sum_{j=1}^{D} \frac{\hat{u}_k (x_j - \hat{v}_j)^2}{\sigma_k^2} (x_j - \hat{v}_j)^2 - \ln \frac{\hat{w}_k}{\hat{u}_k} \alpha_k - \hat{u}_k \ln \alpha_k
\]

If \( (\tilde{Z}, \tilde{\xi}) \) minimizes \( \phi_z(Z, \xi) \), its gradients in the set of variables should vanish. For \( k=1,2,\ldots,K \), we have

\[
\frac{\partial \phi_z(\tilde{Z}, \tilde{\xi})}{\partial \alpha_k} = \sum_{j=1}^{D} \frac{\hat{u}_k (x_j - \hat{v}_j)^2}{\sigma_k^2} = 0,
\]

\[
\frac{\partial \phi_z(\tilde{Z}, \tilde{\xi})}{\partial \hat{w}_k} = \sum_{j=1}^{D} \hat{u}_k (x_j - \hat{v}_j)^2 - \hat{u}_k \ln \alpha_k = 0
\]

and

\[
\sum_{k=1}^{K} \tilde{\alpha}_k = 1.
\]

Eqs. (14) and (15) can be obtained from these. Hence, the result of Theorem 3 follows.

Appendix D Proof of Theorem 4

Proof. When \( U=\tilde{U} \), \( V=\tilde{V} \) and \( Z=\tilde{Z} \) are fixed, the problem of Eq. (11) can be decomposed into \( K \) independent minimization problems:

\[
\min \varphi_k(w_k, \lambda_k) = \lambda_k \left( \sum_{j=1}^{N} w_{kj} - 1 \right) + \sum_{j=1}^{D} \frac{\hat{u}_k (x_j - \hat{v}_j)^2}{\sigma_k^2} (x_j - \hat{v}_j)^2 - \ln w_k
\]

where \( k=1,2,\ldots,K \). If \( (\tilde{w}_k, \tilde{\lambda}_k) \) minimizes \( \varphi_k(w_k, \lambda_k) \), it follows that

\[
\frac{\partial \varphi_k(\hat{w}_k, \hat{\lambda}_k)}{\partial \hat{w}_k} = 0 \quad \text{for} \quad j=1,2,\ldots,D \quad \text{and} \quad \frac{\partial \varphi_k(\hat{w}_k, \hat{\lambda}_k)}{\partial \hat{\lambda}_k} = 0,
\]

yielding

\[
\sum_{j=1}^{D} \hat{w}_k = 1.
\]

Eq. (16) and Eq. (17) can be solved from these two equations, respectively. Hence, the result of Theorem 4 follows.

Appendix E Proof of Theorem 6

Proof. Let \( p \) be an iteration index and \( \hat{U}=U^{(p)} \), \( \hat{V}=V^{(p)} \) and \( \hat{Z}=Z^{(p)} \). According to Theorem 4, it can be seen that \( J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)}) \geq J(U, V, W, Z) \), where \( \hat{W} \) is determined by Eq. (16). Using Eqs. (14) and (16) to replace \( \tilde{\alpha}_k \) and \( \tilde{\omega}_k \) respectively, we obtain

\[
J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)}) \\[ \begin{align*}
& \geq \frac{K}{2} \sum_{k=1}^{K} \frac{\hat{u}_k \sum_{j=1}^{D} \hat{w}_k (x_j - \hat{v}_j)^2}{\sigma_k^2} + N \frac{\hat{u}_k \sum_{j=1}^{D} \hat{w}_k \ln \frac{\hat{w}_k}{\hat{u}_k} \alpha_k - \hat{u}_k \ln \alpha_k}{\sigma_k^2} \\
& \quad - \sum_{k=1}^{K} \hat{u}_k \ln \alpha_k - \sum_{k=1}^{K} \hat{u}_k \ln \frac{1}{\hat{u}_k}
\end{align*}
\\[ \begin{align*}
& \geq \frac{K}{2} \sum_{k=1}^{K} \frac{\hat{u}_k \sum_{j=1}^{D} \hat{w}_k (x_j - \hat{v}_j)^2}{\sigma_k^2} + N \frac{\hat{u}_k \sum_{j=1}^{D} \hat{w}_k \ln \frac{\hat{w}_k}{\hat{u}_k} \alpha_k - \hat{u}_k \ln \alpha_k}{\sigma_k^2} \\
& \quad + \sum_{k=1}^{K} \hat{u}_k \ln \left( 1 - \frac{1}{\hat{u}_k} \right) + \sum_{k=1}^{K} \hat{u}_k (1 - \frac{1}{\hat{u}_k})
\end{align*}
\]

The last step in the above derivation is based on the fact that \( \forall z > 0 : -\ln z \geq 1 - z \). It is assumed that the data are normalized before clustering, i.e., \( 0 \leq x_k \leq 1 \), therefore. On the other hand, according to Theorem 5, \( X_{kv} > 2 \hat{\lambda}_{k} \sigma_k^2 \). In addition, due to \( \sum_{k=1}^{K} \hat{\alpha}_k = 1 \) and \( \hat{\alpha}_k \geq 0 \) for \( k=1,2,\ldots,K \), we have \( \sum_{k=1}^{K} \hat{\alpha}_k \leq 1 \). Based on these facts, the inequality

\[
\sum_{k=1}^{K} \hat{\alpha}_k \leq 1
\]
can be further transformed into
\[ J(U^{(p)}, V^{(p)}, W^{(p)}, Z^{(p)}) \]
\[ \geq N \sum_{k=1}^{K} \frac{N}{2} - N \sum_{k=1}^{K} \frac{2N - N - K}{2} \]
\[ \geq N(1 - K) \]

The result of Theorem 6 follows.

References

[38] L. Jing, M. K. Ng, J. Xu, and J. Z. Huang, "A Text Clustering System Based on k-means Type Subspace Clustering", International Journal of Intelligent Technology, 2006, 1(2):91-


