Fuzzy Co-clustering with Automated Variable Weighting

Charlotte Laclau  
LIPADE - University Paris Descartes  
45 rue des Saint-Peres,  
75006 Paris, France  
charlotte.laclau@parisdescartes.fr

Francisco de A.T. de Carvalho  
Centro de Informatica - CIn/UFPE  
Av. Prof. Luiz Freire, s/n, Cidade Universitaria,  
CEP 50.740-540 Recife, PE, Brazil  
fact@cin.ufpe.br

Mohamed Nadif  
LIPADE - University Paris Descartes  
45 rue des Saint-Peres,  
75006 Paris, France  
mohamed.nadif@parisdescartes.fr

Abstract—We propose two fuzzy co-clustering algorithms based on the double Kmeans algorithm. Fuzzy approaches are known to require more computation time than hard ones but the fuzziness principle allows a description of uncertainties that often appears in real world applications. The first algorithm proposed, fuzzy double Kmeans (FDK) is a fuzzy version of double Kmeans (DK). The second algorithm, weighted fuzzy double Kmeans (W-FDK), is an extension of FDK with automated variable weighting allowing co-clustering and feature selection simultaneously. We illustrate our contribution using Monte Carlo simulations on datasets with different parameters and real datasets commonly used in the co-clustering context.

I. INTRODUCTION

Co-clustering also known as biclustering or block-clustering involves simultaneous clustering of the set of observations and the set of features in a data matrix. By creating permutations of rows and columns, the co-clustering algorithms aim to reorganize the initial data matrix into homogeneous blocks. These blocks, also called co-clusters, can therefore be defined as subsets of the data matrix characterized by a set of observations and a set of features whose elements are similar (as illustrated in figure 1). Other types of co-clustering approaches can be found in [1] and [2].

Co-clustering algorithms present several advantages: they reduce the initial matrix into a simpler form with the same basic structure and require far less computation when compared with separate processing of the same two sets. As a result, these methods are of interest to data mining. In past literature, co-clustering proposals have involved a wide variety of applications. In text mining, several methods exploiting the duality between documents and words exist. For example, a spectral block clustering method [3] and a co-clustering algorithms based on mutual information have been proposed [4], [5]. In gene expression analysis, micro-array experiments result in gene expression matrices whose rows represent genes and whose columns represent various environmental conditions. In [6] the authors formulated this problem by proposing a mean square residue score for measuring cluster quality. This demonstrated the importance of simultaneous clustering for discovering more coherent and meaningful clusters.

In fuzzy co-clustering, observations no longer belong to one particular co-cluster. Instead, there is, for each observation and for each variable, a degree of belonging. As for co-clustering, we can distinguish between partitioning and probabilistic approaches. Probabilistic methods use the framework of mixture model and in [7] the authors proposed block fuzzy c-methods block FCM to deal with binary data. More recently, in [8] the authors proposed a block fuzzy k-modes (block FKM) to handle categorical data. Most of the fuzzy co-clustering algorithms belongs to the family of partitioning methods. Within this family another distinction can be made between partitioning-ranking and dual partitioning algorithms. Most of these algorithms were developed to deal with co-occurrences tables and thus to address the problem of document clustering. Among them, we can cite the FCCM algorithm [9]. In [10] a version of FCCM more suitable for high dimensional data named Fuzzy-CoDok has been proposed; it uses Gini Index instead of the entropy to introduce the fuzziness. However, these algorithms actually perform local feature selection (as subspace clustering [11] algorithms) and not simultaneous clustering of the sets of objects and variables (as co-clustering algorithms). They are basically only looking for the partition of object and the importance of the features in each cluster. In [12], the authors proposed HFCR, a dual partitioning based on fuzzy co-clustering algorithm.

In this article, we propose two distinct fuzzy co-clustering algorithms. The first is a fuzzy version of the well established double Kmeans algorithm [13] for continuous data. A fuzzy version of double Kmeans has been previously proposed in [14]; however in their model, fuzziness is introduced using Gini Index without considering the Khun and Tucker conditions [15] needed to ensure the optimal solution of their algorithm. Therefore, the convergence cannot always be guaranteed. The second algorithm builds upon the first by using an adaptive euclidean distance [16] which allows simultaneous fuzzy co-clustering and feature selection. This is specifically pertinent within a high dimensional context being that feature selection successfully identifies non-relevant variables which negatively impact the performance of clustering algorithms.

The remaining of this paper is organized as follows. Section 2 provides the needed background on the double Kmeans algorithm. Section 3 presents the fuzzy-double Kmeans (FDK) algorithm. This is followed by the explanation of the weighted double Kmeans (W-FDK) algorithm. Section 4 is devoted to numerical experiments on both synthetic and real datasets showing the appropriateness of our contribution. The final section sums up the study and gives recommendations for further research.

Notation. Let $X := \{x_{ij}; i \in I; j \in J\}$ be a data matrix of size $n \times p$ where $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, p\}$. 

The set \( I \) corresponds to the set of \( n \) objects and the set \( J \) to the set of \( p \) continuous attributes. In the sequel, our aim consists in obtaining co-clustering of \( X \). Let \( Z = \{ z_1, \ldots, z_n \} \) be a label vector, where \( z_i \in \{ 1, \ldots, K \} \), denotes the partition of \( I \) into \( K \) clusters and \( W = \{ w_1, \ldots, w_p \} \) where \( w_j \in \{ 1, \ldots, H \} \) denotes the partition of \( J \) into \( H \) clusters. In order to connect hard and fuzzy co-clustering, we also consider \( Z \) and \( W \) as binary classification matrices of size \( n \times K \) and \( p \times H \) respectively and defined by: \( z_{kh} = 1 \) if \( z_i = k \) and \( 0 \) otherwise, and \( w_{jh} = 1 \) if \( w_j = h \) and \( 0 \) otherwise. Finally, to simplify the notation, the sums relating to rows, columns, row clusters and column clusters will be subscripted respectively by the letters \( i = 1, \ldots, n, j = 1, \ldots, p, k = 1, \ldots, K \) and \( h = 1, \ldots, H \), without indicating the implicit limits of variation. For example, the sum \( \sum_{i,k} \) stands for \( \sum_{i=1}^{n} \sum_{k=1}^{K} \).

II. DOUBLE KMEANS ALGORITHM

The co-clustering can be formulated as the search for a good matrix approximation of the original data matrix \( X \). The quality is determined by the approximation error which can be measured by a large class of loss functions like the square Euclidean distances. This approximation is generally achieved through an alternate least square minimization process (see for instance [17]–[19]). The double K-means algorithm [13] (DK) is based on this principle. Formally, the process is to minimize an objective function \( J(Z, W, G) \) where \( Z \) and \( W \) are the partitions and \( G := \{ g_{kh} \}; k \in \{ 1, \ldots, K \}, h \in \{ 1, \ldots, H \} \) is a \( K \times H \) matrix which can be viewed as a summary of the data matrix \( X \) (see figure 2). Each element \( g_{kh} \) of \( G \) is also called a prototype of co-cluster \( X_{kh} := \{ x_{ij} \}; z_{ik}w_{jh} = 1 \}. Double K-means adopts the squared Euclidean distance to measure the dissimilarity between the matrix \( X \) and the structure described in \( Z, W \) and \( G \). Therefore, \( J(Z, W, G) \) is given by

\[
J(Z, W, G) = \sum_{i,k,j,h} z_{ik} \times w_{jh} (x_{ij} - g_{kh})^2
\]  

(1)

It is easy to see that for a fixed \( Z, W \) the optimal values of \( G \) are the means of \( X_{kl} \)'s. The optimal partitions \( Z \) and \( W \) are found using an iterative algorithm. A version of double Kmeans is presented in Algorithm 1 where \( z_k \) (resp. \( w_h \)) represents the cardinality of the \( k^{th} \) cluster (resp. \( h^{th} \) cluster).

III. FUZZY CO-CLUSTERING

In DK we assume that objects and variables belong to one single block or co-cluster. However in some cases, different co-clusters may overlap each other, and hence objects and variables may have multiple memberships. This leads to a fuzzy version of DK called fuzzy double Kmeans (FDK).

A. Objective function of FDK

The FDK algorithm optimizes an adequacy criterion \( J \) measuring the fit between the co-clusters and their prototypes.

\[
J_F(U, V, G) = \sum_{i,k,j,h} u_{ik}^\beta v_{jh}^\alpha d_{ijkh}^2
\]  

(2)

in which \( d_{ijkh} = |x_{ij} - g_{kh}| \) is the distance from datapoints to prototypes. Furthermore, \( U := \{ u_{ik}: i = 1, \ldots, n; k = 1, \ldots, K \} \) where \( u_{ik} \) denotes the membership degree of object \( i \) in \( k^{th} \) cluster and \( V := \{ v_{jh}: j = 1, \ldots, p; h = 1, \ldots, H \} \) where \( v_{jh} \) denote the membership degree of variable \( j \) in \( h^{th} \) cluster. \( \alpha \in (1, \infty) \) and \( \beta \in (1, \infty) \) are the parameters that control the fuzziness of membership for each object and each variable respectively. The larger they are, the fuzzier
the resulting co-clusters will be. The objective function is minimized subject to the following constraints:

\[ \sum_k u_{ik} = 1, u_{ik} \in [0, 1] \quad (3) \]

and

\[ \sum_h v_{jh} = 1, v_{jh} \in [0, 1]. \quad (4) \]

To minimize the criterion \( J_F \) defined in equation 2 with respect to the necessary conditions for optimality given by equation 3 and equation 4 we use the method of Lagrange multipliers and obtain

\[ J_F(U, V, G) = \sum_{i,j,k,h} u_{ik}^\alpha v_{jh}^\beta d_{ijkh}^2 - \theta_1 \left( \sum_k u_{ik} - 1 \right) - \theta_2 \left( \sum_h v_{jh} - 1 \right) \]

where \( \theta_1 \) and \( \theta_2 \) are the Lagrange multipliers.

**B. The FDK algorithm**

The algorithm iterates three steps until the convergence of \( J_F \) to a local minimum. These steps are described below.

1. The first step consists in the computation of the matrix of prototypes \( G \). With \( U \) and \( V \) fixed, the minimization of \( J_F \) is obtained when \( \forall k, h \)

\[ g_{kh} = \frac{\sum_{i,j} (u_{ik})^\alpha (v_{jh})^\beta x_{ij}}{\sum_{i,j} (u_{ik})^\alpha (v_{jh})^\beta}. \quad (5) \]

2. In the second step, given \( G \) and \( V \) we update \( U \), the data matrix of membership degrees for each object. Let \( A = \{ k \in \{ 1, \ldots, K \} : D_{ik} = 0 \} \) where

\[ D_{ik} = \sum_{j,h} (v_{jh})^\beta d_{ijkh}^2 \quad \forall i, k. \quad (6) \]

- if \( A = \emptyset \) i.e. no object coincides with any of the representatives, by setting the derivatives of \( J_F \) with respect to \( u_{ik} \) and \( \theta_1 \) to zero, we obtain:

\[ u_{ik} = \left( \sum_{r=1}^K (D_{ik}/D_{ir})^{1/\gamma} \right)^{-1} \quad \forall i, k. \quad (7) \]

- if \( A \neq \emptyset \) then [20] 

\[ \left\{ \begin{array}{l} u_{ik} = 1/|A|, \forall k \in A \\ u_{ir} = 0, \forall r \notin A \end{array} \right. \]

where \( |A| \) denotes the cardinality of \( A \).

3. In the last step, assuming that \( G \) and \( U \) are fixed, we update \( V \) the data matrix of membership degrees for each variable using a similar assignment process than for \( U \). Let \( B = \{ h \in \{ 1, \ldots, H \} : D_{jh} = 0 \} \) where

\[ D_{jh} = \sum_{i,k} (u_{ik})^\alpha d_{ijkh}^2 \quad \forall j, h. \quad (8) \]

- if \( B = \emptyset \), by setting the derivatives of \( J \) with respect to \( v_{jh} \) and \( \theta_2 \) to zero, we obtain

\[ v_{jh} = \left( \sum_{s=1}^H (D'_{jsh}/D'_{js})^{1/\gamma} \right)^{-1} \quad \forall j, h. \quad (9) \]

- if \( B \neq \emptyset \) then 

\[ \left\{ \begin{array}{l} v_{jh} = 1/|B|, \forall h \in B \\ v_{js} = 0, \forall s \notin B \end{array} \right. \]

where \( |B| \) denotes the cardinality of \( B \).

We repeat the three steps until the convergence is obtained i.e. the \( J_F \) value change is small or there is no change. These steps are summarized in Algorithm 2.

**Algorithm 2 Fuzzy Double Kmeans (FDK)**

**input:** \( X, K, H, \alpha \) and \( \beta \)

**initialization:** random initialization of \( U \) and \( V \)

**repeat**

(1) Compute \( G \) using equation 5

(2) Update \( U \) using equation 7

(3) Update \( V \) using equation 9

**until** the \( J_F \) value change is small or there is no change

**output:** \( G, U, V \)

IV. FUZZY CO-CLUSTERING WITH VARIABLE WEIGHTING

In this section, we propose the weighted fuzzy double Kmeans (W-FDK) algorithm which is able to differentiate relevant variables from noisy ones. This algorithm is an extension of the FDK algorithm described in the previous section, with an additional step to calculate the weights of variables.

**A. Objective function of W-FDK**

As a result of the use of an adaptive distance, variable weights are automatically calculated within the minimization process. The new objective function is defined as:

\[ J_W(U, V, \Lambda, G) = \sum_{i,j,k,h} u_{ik}^\alpha v_{jh}^\beta \delta_{ijkh} \quad (10) \]

where \( \delta_{ijkh} = \lambda_j^{-1} d_{ijkh}^2 \) is a squared weighted euclidean distance. Furthermore, \( \Lambda := \{ \lambda_j : j = 1, \ldots, p \} \) with \( \lambda_j \) that represents the weight of the variable \( j \), \( \gamma \in (1, \infty) \). The optimization of \( J_W \) is subject to the following constraint:

\[ \left\{ \begin{array}{l} \sum_k u_{ik} = 1, u_{ik} \in [0, 1] \\ \sum_h v_{jh} = 1, v_{jh} \in [0, 1] \\ \sum_j \lambda_j = 1, \lambda_j \in [0, 1] \end{array} \right. \quad (11) \]

Therefore, the minimization of \( J_W \) with respect to \( U, V \) and \( \Lambda \), subject to the constraint given in equation 11, leads to the following Lagrangian function:

\[ J_W(U, V, G) = \sum_{i,j,k,h} u_{ik}^\alpha v_{jh}^\beta d_{ijkh}^2 - \sum_i \theta_1 \left( \sum_k u_{ik} - 1 \right) - \sum_j \theta_2 \left( \sum_h v_{jh} - 1 \right) - \sum_j \theta_3 \left( \sum_k \lambda_j - 1 \right). \]

**B. Weighted Fuzzy Double Kmeans algorithm**

The algorithm iterates the four steps described below until the convergence of the criterion given in equation 10.
In the same way as for the FDK algorithm: given the current \( U, V \) and \( \Lambda \), the minimizer of \( G \) is defined by \( \forall k, h \)

\[
g_{kh} = \frac{\sum_{i,j} (u_{ik})^\alpha (v_{kj})^\beta \lambda_j}{\sum_{i,j} (u_{ik})^\alpha (v_{kj})^\beta \lambda_j^2} \cdot x_{ij}. \tag{12}
\]

(2) This step consists in the computation of the weights of variables. Let \( C = \{ j \in \{1, \ldots, p\} : D''_j \neq 0 \} \) where \( \forall j \)

\[
D''_j = \sum_{i,k,h} u_{ik}^\alpha v_{kj}^\beta d_{ijkh}^2.
\]

\( D''_j = 0 \) means that the variable \( j \) has the same value in each cluster. This represents a degenerate solution. We adapted the solution proposed by [21] and obtained the following assignment process:

\[
\forall j \quad \begin{cases} 
\lambda_j = \left[ \sum_{w=1}^{|C|} \frac{(D''_j / D''_w)}{\lambda_j^{-1}} \right]^{-1}, & \text{if } j \in C \\
\lambda_j = 0, & \text{if } j \notin C
\end{cases} \tag{13}
\]

where \( |C| \) denotes the cardinality of \( C \). This solution is equivalent to replacing the constraint \( \sum_{j=1}^p \lambda_j = 1 \) by \( \sum_{j=1}^{|C|} \lambda_j = 1 \). When \( j \in C \), the weight of a variable is related to its variance.

(3) This step is equivalent to the step 2 of the FDK algorithm but the formula of \( D \) (see equation 6) has to take into account the information, given by the vector of weights \( \Lambda \), and thus becomes \( \forall i, k \)

\[
D_{ik} = \sum_{j,h} (v_{jk})^\beta (\lambda_j)^\gamma d_{ijkh}^2. \tag{14}
\]

(4) In the same way as for \( U \), we follow the same process assignment than in the step 3 of the FDK algorithm. The \( D''_{jh} \) defined in equation 8 becomes \( \forall j, h \)

\[
D''_{jh} = \sum_{i,k} u_{ik}^\alpha (\lambda_j)^\gamma d_{ijkh}^2. \tag{15}
\]

These steps are summarized in Algorithm 3.

**Algorithm 3** Fuzzy Weighted Double Kmeans (W-FDK)

**input:** \( X, K, H, \alpha, \beta \) and \( \gamma \)  
**initialization:** random initialise of \( U, V \) and \( \Lambda \)  
**repeat**  
(1) Compute \( G \) using equation 12  
(2) Compute \( \Lambda \) using equation 13  
(3) Update \( U \) using equation 7 based on 14  
(4) Update \( V \) using equation 9 based on 15  
**until** the \( J_W \) value change is small or there is no change  
**output:** \( G, \Lambda, U, V \)

**C. Illustration of W-FDK**

In this subsection we aim to illustrate the behavior of W-FDK on a simple example. Let’s consider a data matrix \( X \) of size \( 8 \times 8 \) where 2 of the features are noise (in blue). The number of clusters for row and the number of clusters for the columns are equal to 2.

\[
\begin{array}{cccccc}
0.9 & 2.9 & 2.9 & 3.0 & 3.0 & 2.9 \\
0.9 & 2.9 & 2.9 & 3.0 & 3.0 & 2.9 \\
2.8 & 0.9 & 1.0 & 1.1 & 1.0 & 0.9 \\
3.0 & 1.0 & 0.9 & 1.0 & 0.9 & 0.5 \\
2.8 & 0.9 & 0.9 & 1.1 & 0.9 & 1.0 \\
0.9 & 3.0 & 2.9 & 3.0 & 3.1 & 2.9 \\
2.9 & 0.8 & 0.9 & 1.0 & 1.1 & 1.0 \\
1.1 & 3.0 & 2.9 & 3.0 & 3.1 & 3.2 \\
\end{array}
\]

Hereafter, we present the same data matrix without the two noisy features and reorganized according to the partition of rows and the partition of columns. Therefore, it becomes easy to reveal the new structure of the data matrix reorganized into homogeneous blocks.

\[
\begin{array}{cccccc}
0.9 & 2.9 & 2.9 & 3.0 & 3.0 & 2.9 \\
0.9 & 2.9 & 2.9 & 3.0 & 3.0 & 2.9 \\
0.9 & 3.0 & 2.9 & 3.0 & 3.1 & 2.9 \\
1.1 & 3.0 & 2.9 & 3.0 & 3.1 & 3.1 \\
2.8 & 0.9 & 0.9 & 1.1 & 0.9 & 0.9 \\
3.0 & 1.0 & 0.9 & 1.0 & 0.9 & 0.9 \\
2.8 & 0.9 & 0.9 & 1.1 & 0.9 & 1.0 \\
2.9 & 0.8 & 0.9 & 1.0 & 1.1 & 1.0 \\
\end{array}
\]

In the sequel, we describe the main steps of W-FDK allowing to obtain the partitioning of \( X \) while selecting the relevant features. As usual, the data are normalized before starting the algorithm.

**Input:** we set the parameters \( K \) and \( H \) to be the real number of row and column clusters i.e. 2.

**Initialization:** we randomly initialize the membership matrix \( U \) and \( V \) and the vector of feature weights \( \Lambda \) with respect to the constraint given by equations 11. We have

\[
\begin{array}{cccc}
0.80 & 0.20 & 0.94 & 0.06 \\
0.78 & 0.22 & 0.80 & 0.20 \\
0.75 & 0.25 & 0.46 & 0.54 \\
0.14 & 0.86 & 0.18 & 0.82 \\
0.80 & 0.20 & 0.82 & 0.18 \\
0.28 & 0.72 & 0.69 & 0.31 \\
0.12 & 0.88 & 0.55 & 0.45 \\
0.58 & 0.42 & 0.50 & 0.50 \\
\end{array}
\]

\[
\begin{array}{cccc}
0.00 & 0.13 & 0.12 & 0.12 \\
0.13 & 0.12 & 0.12 & 0.13 \\
0.14 & 0.15 & 0.14 & 0.15 \\
\end{array}
\]

**Step 1:** \( U^0, V^0 \) and \( \Lambda^0 \) are fixed. We first compute the matrix of prototypes using equation 12 and obtained the following result:

\[
G^0 = \begin{pmatrix}
0.11 & 0.18 \\
-0.16 & -0.27
\end{pmatrix}
\]

**Step 2:** \( G^0, U^0 \) and \( V^0 \) are fixed. We update the vector of feature weights using the process described in the previous subsection in step (2). This leads to

\[
\begin{pmatrix}
0.09 & 0.12 & 0.13 & 0.12 & 0.12 & 0.13 & 0.14 & 0.15
\end{pmatrix}^T
\]

Starting from random \( U \) and \( V \), the first estimates of feature weights give each feature an importance approximately equal.

**Step 3 and 4:** in order to update the membership matrix \( U \) (respectively \( V \)), we fix \( G^0, V^0 \) and \( \Lambda^0 \) (respectively \( G^0, U^1 \) and \( \Lambda^1 \)) and following the assignment process described.
in the previous subsection, step 3 (respectively step 4) we obtain successively:
\[
U^1 = \begin{pmatrix} 0.74 & 0.26 \\ 0.71 & 0.29 \\ 0.51 & 0.49 \\ 0.21 & 0.79 \\ 0.27 & 0.73 \\ 0.60 & 0.40 \\ 0.22 & 0.80 \\ 0.78 & 0.22 \end{pmatrix} \quad \text{and} \quad V^1 = \begin{pmatrix} 0.56 & 0.44 \\ 0.46 & 0.54 \\ 0.46 & 0.54 \\ 0.46 & 0.54 \\ 0.46 & 0.54 \\ 0.47 & 0.53 \\ 0.52 & 0.48 \end{pmatrix}
\]

Finally, we compute the value of the criterion \( J_W \) (see equation 10). After iteration 1, its value is 0.42. We repeat the steps 2 to 4 until convergence. The convergence is achieved after 12 iterations.

At the end, we obtain the following feature weight vector:
\[
\Lambda^{12} = (0.03 \ 0.19 \ 0.23 \ 0.22 \ 0.17 \ 0.16 \ 0.00 \ 0.00)^T
\]
We notice that the first six features have higher weight than the two irrelevant ones which are close to zero. As regards the membership matrices of objects and features, we have
\[
U^{12} = \begin{pmatrix} 1.00 & 0.00 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \\ 0.00 & 1.00 \\ 1.00 & 1.00 \\ 0.00 & 1.00 \\ 1.00 & 1.00 \\ 1.00 & 0.00 \end{pmatrix}, \quad V^{12} = \begin{pmatrix} 0.99 & 0.01 \\ 0.00 & 1.00 \\ 0.00 & 1.00 \\ 0.00 & 1.00 \\ 0.00 & 1.00 \\ 0.19 & 0.81 \\ 0.91 & 0.09 \end{pmatrix}
\]

From matrices \( U \) and \( V \) we can deduce \( Z \) and \( W \), the hard partition of objects and variables respectively. This is achieved by using the maximum a posteriori (MAP) principle. Therefore we obtain the following partitions:
\[
Z = (1 \ 1 \ 2 \ 2 \ 2 \ 1 \ 2 \ 1)^T \quad \text{and} \quad W = (1 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 \ 1)^T.
\]

The last value of the criterion \( J_W \) is 0.016 and \( W-FDK \) leads to the correct co-clustering.

With \( FDK \), using the same initialization than for \( W-FDK \), we obtain the convergence after 21 iterations.
\[
U^{21} = \begin{pmatrix} 1.00 & 0.00 \\ 1.00 & 0.00 \\ 0.02 & 0.98 \\ 0.03 & 0.97 \\ 0.03 & 0.97 \\ 1.00 & 0.00 \\ 0.03 & 0.97 \\ 1.00 & 0.00 \end{pmatrix}, \quad V^{21} = \begin{pmatrix} 0.21 & 0.79 \\ 1.00 & 0.00 \\ 1.00 & 0.00 \\ 1.00 & 0.00 \\ 1.00 & 0.00 \\ 0.00 & 1.00 \\ 0.00 & 1.00 \end{pmatrix}
\]

The resulting membership matrices, similar than \( DK \), are:
\[
Z = (1 \ 1 \ 2 \ 2 \ 2 \ 1 \ 2 \ 1)^T \quad \text{and} \quad W = (1 \ 2 \ 2 \ 2 \ 2 \ 1 \ 1 \ 1)^T.
\]

For these three metrics, a value close to 1 means a good clustering result.

\section*{V. Numerical Experiments}
This section is devoted to the evaluation of the proposed algorithms on both synthetic and real datasets.

\subsection*{A. Performance evaluation}
In order to assess and to compare the performance of the proposed algorithms we use the commonly adopted metrics, the accuracy, the Normalize Mutual Information [22] and the Adjusted Rand Index [23]. We focus only on the quality of row clustering. Clustering accuracy noted Acc discovers the one-to-one relationship between clusters and classes and measures the extent to which each cluster contained data points from the corresponding class; it is defined as follows:
\[
Acc = \frac{1}{n} \max \left[ \sum_{C_k, L_\ell} T(C_k, L_\ell) \right]
\]
where \( C_k \) is the \( k \)th cluster in the final results, and \( L_\ell \) is the true \( \ell \)th class. \( T(C_k, L_\ell) \) is the number of entities which belong to class \( m \) and are assigned to cluster \( k \). Accuracy computes the maximum sum of \( T(C_k, L_\ell) \) for all pairs of clusters and classes, and these pairs have no overlaps. The second measure employed is the Normalized Mutual Information (NMI); it is estimated by
\[
\text{NMI} = \frac{\sum_{k,\ell} n_k \log \frac{n_k}{n}}{\sqrt{(\sum_{k} n_k \log \frac{n_k}{n})(\sum_{\ell} n_\ell \log \frac{n_\ell}{n})}}
\]
where \( n_k \) denotes the number of data contained in the cluster \( C_k (1 \leq k \leq K) \), \( n_\ell \) is the number of data belonging to the class \( L_\ell (1 \leq \ell \leq K) \), and \( n_{k\ell} \) denotes the number of data that are in the intersection between the cluster \( C_k \) and the class \( L_\ell \).

The last measure is the Adjusted Rand Index noted ARand is a measure of the similarity between two data clustering partitions. From a mathematical standpoint, the Rand index is related to the accuracy. The adjusted form of the Rand Index is:
\[
\text{ARand} = \frac{\sum_{k,\ell} (n_{k\ell}) - \left[ \sum_{k} (n_k) \sum_{\ell} (n_\ell) \right] / (n)}{\frac{1}{2} \left[ \sum_{k} (n_k) + \sum_{\ell} (n_\ell) \right] - \left[ \sum_{k} (n_k) \sum_{\ell} (n_\ell) \right] / (n)}.
\]

Using the latent block model proposed in [24] and extended in the continuous situation [19] and, assuming that for each co-cluster \( X_{kh} \) the values \( z_{ij} \) are distributed according to a Gaussian distribution \( \mathcal{N}(\mu_{kh}, \sigma_{zh}^2) \) with \( \mu_{kh} \in \mathbb{R} \) and \( \sigma_{zh}^2 \in \mathbb{R}^+ \), we obtain the Gaussian latent block model with the following probability density function taking this form
\[
f(x; \theta) = \sum_{(Z,W)} \prod_{i,k} \pi_{z_{ik}}^{z_{ik}} \prod_{j,\ell} \mu_{\ell}^{w_{\ell j}} \prod_{i,j,k,h} \phi(x_{ij}; \mu_{kh}, \sigma_{zk}^2)
\]
This model is then parametrised by

\[ \varphi(x_{ij}; (\mu_{kh}, \sigma_{kh}^2)) = \frac{1}{\sqrt{2\pi\sigma_{kh}^2}} \exp\left\{ \frac{(x_{ij} - \mu_{kh})^2}{2\sigma_{kh}^2} \right\} . \]

This model is then parametrised by \( \theta = (\pi, \rho, \delta) \) where \( \pi = (\pi_1, \ldots, \pi_K) \), \( \rho = (\rho_1, \ldots, \rho_H) \) and \( \delta = ((\mu_1, \sigma_{11}^2), \ldots, (\mu_KH, \sigma_{KH}^2)) \).

Algorithm 4 Simulation of data

**input:** \( n, p_1 \) relevant and \( p_2 \) irrelevant attributes \((p_1 + p_2 = p)\), \( H \), \( K \) and \( \delta \). 

**generate a data matrix \( X_1 \) of size \( n \times p_1 \):** 
(1) Simulate \( Z \) according to a multinomial distribution with parameters \((1, \pi_1, \ldots, \pi_K)\) 
(2) Simulate \( W \) according to a multinomial distribution with parameters \((1, \rho_1, \ldots, \rho_H)\) 
(3) Simulate each co-cluster \( X_{kh} \) according to Gaussian density with \((\mu_{kh}, \sigma_{kh}^2)\)

**generate a data matrix \( X_2 \) of size \( n \times p_2 \):** 
(4) Simulate the \( p_2 \) irrelevant variables according to Gaussian distribution \( N((\mu_{p_1+1}, \ldots, \mu_{p_2+1}, \sigma^2 I)) \)

**output:** data matrix \( X = (X_1, X_2) \) of size \((n \times (p_1 + p_2))\)

C. Synthetic datasets

We used synthetic data in order to evaluate our methods FDK and W-FDK, and compare the results with those of Kmeans and DK since these two algorithms minimize an objective function using the same distance. We simulated continuous datasets with varying size, degree of block overlap and assuming equal proportions of the clusters i.e. \( \pi_1 = \ldots = \pi_K \) and \( \rho_1 = \ldots = \rho_H \). For each dataset we took \( K = H = 5 \). Furthermore, to demonstrate the impact of irrelevant variables on the performance of the algorithms without feature selection, we run the experiments on the same datasets after introducing 20% of noise to each one. Regarding to the size, we took \( n \times p = 1000 \times 500, 1000 \times 1000 \). Figure 3 shows data matrices with 500 features, 1000 objects, equal proportions and two different degrees of overlapping. We can clearly see the block structure of the data as well as the irrelevant features. For each data structure, we generate 200 samples. For each sample we ran the algorithms 30 times. In tables I and II are reported the averages and standard deviations of Acc’s, NMI’s and ARand’s for each situation. Several observations can be made based on these results.

- Both proposed algorithms, FDK and W-FDK converge i.e. the criterion is minimized in both cases as illustrated in figures 4 (a) and (b). Furthermore it can be noted that W-FDK converges more quickly than FDK with an equal degree of fuzziness. Indeed, in this specific case, the FDK stops after 27 iterations while W-FDK only needs 10 iterations. In all experiments conducted here, we observe that generally W-FDK requires about half of the number of iterations required by FDK.

- When datasets consist of well separated block clusters (+) and without noise, DK, FDK and W-FDK perform equally well. Note that FDK always gives slightly better solutions. However, when the degree of overlapping is higher (+++) the fuzzy algorithms FDK and W-FDK outperform DK. The Kmeans algorithm is always outperformed.

- In table I and II, it appears that no matter the number of features, all algorithms except the Kmeans have the same and good behaviors. Indeed, for dataset with size \( 1000 \times 500 \) and \( 1000 \times 1000 \), we obtain similar Acc’s, NMI’s and ARI’s. This result is not very surprising since co-clustering algorithms are well known for their ability to deal with large datasets.

- On datasets where noisy features were introduced (table II), the fuzzy algorithm with feature selection W-FDK always gives the best results regardless the size or the degree of overlapping. From figure 4 (c), we can see that W-FDK is able to identify the relevant features. Indeed, the first 400 features have significantly higher weight than the last 100 which are in this case the noisy features.

D. Real datasets

Experiments are conducted for 3 datasets. CSTR contains 476 documents described by 1000 words. Originally each
classes on both datasets. The number of feature clusters is set

TABLE III: Datasets Description (# denotes the cardinality).

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#objects</th>
<th>#features</th>
<th>#clusters</th>
<th>Data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>COIL.20</td>
<td>1440</td>
<td>1024</td>
<td>20</td>
<td>Image</td>
</tr>
<tr>
<td>CSTR</td>
<td>475</td>
<td>1000</td>
<td>4</td>
<td>Document</td>
</tr>
<tr>
<td>Yale</td>
<td>165</td>
<td>1024</td>
<td>15</td>
<td>Image</td>
</tr>
</tbody>
</table>

parameter settings and select the best result for each one ac-

TABLE II: Means of Acc, NMI and ARand (± standard errors) computed on 200 samples where noisy features were introduced.

cell of this dataset denotes the number of occurrences of a
word in a document. As we are interested in algorithms for
continuous data we use a version of CSTR on which a TF-
IDF transformation was first applied. The COIL.20 is an image
database, it contains 1440 objects and 1024 features. The Yale
Face Database contains 165 grayscale images of 15 individuals
described by 1024 features. These datasets are described in

Because each clustering algorithm has one or more par-

Fig. 4: Behavior of the criterion $J_F$ for FDK (a) and $J_W$ for W-FDK (b). (c) represents the feature weight obtained by W-FDK. These results are obtained on a dataset of size $1000 \times 500$ with noise and a high degree of overlap (++).
Adjusted Rand Index obtained on real datasets.

TABLE IV: Accuracy, Normalized Mutual Information and criterion of all trials for each one in table IV.

We can see from table IV that on these three datasets, both FDK and W-FDK outperform Kmeans and DK. On COIL20, FDK is slightly better than W-FDK. However, for the CSTR dataset, W-FDK gives notably better results than FDK. On the Yale dataset, both algorithms perform equally well. About the number of iterations, figure 5 shows that for the three datasets W-FDK requires far less iterations than FDK.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Metrics</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Kmeans</td>
</tr>
<tr>
<td>CSTR</td>
<td>Acc</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td>NMI</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td>ARand</td>
<td>0.40</td>
</tr>
<tr>
<td>COIL20</td>
<td>Acc</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td>NMI</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>ARand</td>
<td>0.53</td>
</tr>
<tr>
<td>Yale</td>
<td>Acc</td>
<td>0.53</td>
</tr>
<tr>
<td></td>
<td>NMI</td>
<td>0.53</td>
</tr>
<tr>
<td></td>
<td>ARand</td>
<td>0.28</td>
</tr>
</tbody>
</table>

TABLE IV: Accuracy, Normalized Mutual Information and Adjusted Rand Index obtained on real datasets.

VI. CONCLUSION AND FUTURE WORK IN PROGRESS

In this paper we presented a fuzzy version of double Kmeans noted FDK and a extension of FDK named W-FDK which can automatically compute the weights of variables and thus performs feature selection. Compared with other methods, we demonstrate that our proposed algorithms are more efficient in terms of co-clustering. In addition, the W-FDK offers a suitable manner for feature selection and requires a smaller number of iterations than FDK.

For further research, it will be worthwhile to investigate an efficient way to choose the value of the parameter $\alpha$, $\beta$ and $\gamma$ and to study in more details its impact on the clustering performance.

REFERENCES