# Fuzzy Co-Clustering and Kernel Fuzzy Co-Clustering for Interval-Valued Data

José Nataniel A. de Sá
Universidade Federal de Pernambuco
Centro de Informática - CIn/UFPE
E-mail: jnas@cin.ufpe.br

Marcelo R.P. Ferreira
Universidade Federal da Paraíba
Departamento de Estatística - DE/UFPB
E-mail: marcelo@de.ufpb.br

Francisco de A.T. de Carvalho
Universidade Federal de Pernambuco
Centro de Informática - CIn/UFPE
E-mail: fatc@cin.ufpe.br

***Abstract*—Interval-valued data are a type of Symbolic Data, a field of statistics and artificial intelligence that deals with multivalued data representations. Such representations arise from the need to preserve information about data variability during data aggregation processes, such as aggregating a continuous feature according to a categorical attribute that often uses summary statistics like the mean or median. This paper proposes co-clustering algorithms for interval-valued data. These algorithms simultaneously group samples and features and have demonstrated competitive performance in several applications. We introduced one variant without kernel functions and two variants employing the Gaussian kernel, making them suitable for grouping nonlinear clusters. Experiments conducted on synthetic and benchmark interval-valued datasets demonstrated the effectiveness of the proposed algorithms, particularly the kernel-based variants.**

***Index Terms*—Interval-valued data, Co-clustering, Gaussian kernel.**

## I. Introduction

Symbolic Data Analysis (SDA) is a field of statistics and artificial intelligence that focuses on the analysis of multivalued data, such as intervals, histograms, and categories [1]. This type of data arises when the objective is to analyze groups of individuals rather than single observations. For instance, while data may be collected at the level of students, the analytical goal may be to study universities. Traditional aggregation methods, such as using the mean or median for numerical features and the mode for categorical features, often lead to a loss of information about data variability. Symbolic data emerges as an alternative to these conventional strategies for better preserving variability, enabling a more informative representation of aggregated data. The focus of this paper is to study interval-valued data.

Clustering is a well-known area of unsupervised learning whose goal is to identify clusters of similar samples. Among clustering algorithms, kernel-based methods are particularly suited to handling nonlinear data by transforming the original data space into a higher-dimensional kernel space. This transformation mitigates a major limitation of two of the most widely used clustering algorithms, K-means and Fuzzy C-Means, which often fail to correctly identify non-spherical and nonlinearly separable clusters.

Traditional clustering methods group only the samples, assume no cluster structure in the features, and implicitly assume that all clusters are defined across all features. In contrast, co-clustering algorithms simultaneously group samples and features based on the associations between them, forming data blocks in which samples and features are correlated. By grouping data in two dimensions, these methods provide a partition of the samples, as in traditional clustering, while also yielding a partition of the features, and have demonstrated competitive performance compared to classical clustering techniques. Co-clustering methods have been successfully applied across several domains, including recommender systems [2], document clustering [3], and gene expression analysis [4].

Kernel-based co-clustering algorithms have also been proposed in the literature. Ref. [5] introduced the first approach combining co-clustering with kernel functions, employing a Gaussian kernel and proposing an algorithm that computes feature weights for sample clusters. Subsequently, Ref. [6] presented hard and fuzzy Gaussian kernel co-clustering methods with automatic computation of the kernel width parameter. Finally, Ref. [7] proposed a method that computes weights for both samples and features, demonstrating strong performance in scenarios with noise in both dimensions.

Despite the broad applicability of co-clustering algorithms, to the best of our knowledge, no existing work has addressed co-clustering methods for interval-valued data. In this context, this article proposes novel co-clustering models for interval-valued data. We introduce three variants: one without kernel functions and two kernel-based variants using the Gaussian kernel. Furthermore, we adopt a fuzzy framework that allows samples and features to belong to multiple clusters with membership degrees ranging from 0 to 1, which is effective in scenarios with overlapping clusters. The remainder of this paper is organized as follows. Section II reviews the basic theory of kernel functions. Section III presents the proposed algorithms. Section IV reports the experimental results. Finally, conclusions and directions for future work are discussed in Section V. The source code and datasets are available at: https://github.com/Natandradesa/fuzzy-coclustering-interval-data.

**Remark**. Due to space limitations, we simplify the summations by indicating only the varying indices, with the following fixed ranges: $i = 1, \ldots, I$, $d = 1, \ldots, D$, $o = 1, \ldots, O$, and $v = 1, \ldots, V$. For example, $\sum_{i,d}$ denotes $\sum_{i=1}^{I} \sum_{d=1}^{D}$.

**Definition** (Interval-valued data). An interval-valued data

point is defined as a realization $\varsigma = [a, b]$, where $a, b \in \mathbb{R}$ and $a \leq b$. Let $\mathbf{X} = [x_{id}]_{\substack{1\leq i\leq I \\ 1\leq d\leq D}}$ denote an interval-valued data matrix composed of $I$ samples described by $D$ interval-valued features. Each entry $x_{id}$ is an interval of the form $x_{id} = [x_{id}^L, x_{id}^U]$, with $x_{id}^L \leq x_{id}^U$, where $x_{id}^L$ and $x_{id}^U$ represent the lower and upper bounds, respectively, of the $d$-th feature for the $i$-th sample. An interval-valued data matrix $\mathbf{X}$ can also be defined as set of $I$ samples $\{\mathbf{x}_1, ..., \mathbf{x}_i, ...\mathbf{x}_I\}$, where the $i$-th sample is described by a vector of intervals $\mathbf{x}_i = \left[(x_{i1}^L, x_{i1}^U), ..., (x_{id}^L, x_{id}^U), ..., (x_{iD}^L, x_{iD}^U)\right]$.

## II. Kernel functions theory

Kernel-based algorithms have been widely developed due to their ability to group nonlinear clusters. These methods are based on kernel functions [8], which map the original data into a higher-dimensional space, commonly referred to as the kernel space, by means of a nonlinear transformation $\Omega$. The main idea behind this mapping is to obtain a more suitable cluster structure in the kernel space; that is, the clustering task is expected to be easier in this space, as the clusters are expected to be better defined.

The applicability of kernel methods is largely due to the *kernel trick* [9]. Using the kernel trick, the mapping $\Omega$ does not need to be known, since the inner product in kernel space can be calculated directly in input space using a kernel function $\mathcal{K}$, i.e $\mathcal{K}(\mathbf{x}_r, \mathbf{x}_s) = \Omega(\mathbf{x}_r)^\top \Omega(\mathbf{x_s})$. Based on this, the Euclidean distances in the kernel space can be calculated as:

$$\begin{aligned}\|\Omega(\mathbf{x}_r) - \Omega(\mathbf{x}_s)\|^2 &= \left[\Omega(\mathbf{x}_r) - \Omega(\mathbf{x}_s)\right]^\top \left[\Omega(\mathbf{x}_r) - \Omega(\mathbf{x}_s)\right] \\ &= \Omega(\mathbf{x}_r)^\top \Omega(\mathbf{x}_r) - 2\Omega(\mathbf{x}_r)^\top \Omega(\mathbf{x}_s) + \Omega(\mathbf{x}_s)^\top \Omega(\mathbf{x}_s) \\ &= \mathcal{K}(\mathbf{x}_r, \mathbf{x}_r) - 2\mathcal{K}(\mathbf{x}_r, \mathbf{x}_s) + \mathcal{K}(\mathbf{x}_s, \mathbf{x}_s) \end{aligned} \quad (1)$$

Kernel-based methods have been developed under two main approaches [10]: clustering in the original space, in which the cluster prototypes (representative values) are defined in the input space and the mapping to the kernel space is performed implicitly, and clustering in the kernel space, where the prototypes are defined directly in the kernel space. In the latter case, distances are computed in the original space using Eq. (1).

A common choice for the kernel function is the Gaussian kernel. In the co-clustering framework, it is necessary to define an appropriate kernel, since distances are computed between data points rather than between vectors. To this end, we rely on the definition presented in Ref. [5] for conventional (real-valued) data. Accordingly, the Interval Gaussian kernel for the co-clustering framework is defined as

$$\mathcal{K}(x_{rj}, x_{sl}) = \exp\left\{\frac{-\left[(x_{rj}^L - x_{sl}^L)^2 + (x_{rj}^U - x_{sl}^U)^2\right]}{2\sigma^2}\right\}, \quad (2)$$

where $\sigma^2$ ($\sigma > 0$) is the width hyperparameter. Based on the Gaussian kernel, and considering the distance in Eq. (1), we have:

$$\|\Omega(x_{rj}) - \Omega(x_{sl})\|^2 = 2 - 2\mathcal{K}(x_{rj}, x_{sl}) \quad (3)$$

## III. Co-clustering algorithms for interval-valued data

This section presents the main contribution of this paper: *fuzzy co-clustering methods for interval-valued data*. Unlike traditional clustering methods, which group only samples, co-clustering algorithms simultaneously group samples and features, providing clusters of samples associated with clusters of features, and vice versa. Accordingly, the proposed co-clustering algorithms produce a fuzzy partition of $I$ samples into $O$ clusters, represented by the matrix $\mathbf{G} = [g_{io}]_{\substack{1\leq i\leq I \\ 1\leq o\leq O}}$, subject to $g_{io} \in [0, 1]$ and $\sum_{o=1}^{O} g_{io} = 1$. In parallel, they also produce a fuzzy partition of $D$ features into $V$ clusters, represented by $\mathbf{H} = [h_{dv}]_{\substack{1\leq d\leq D \\ 1\leq v\leq V}}$, subject to $h_{dv} \in [0, 1]$ and $\sum_{v=1}^{V} h_{dv} = 1$.

By simultaneously grouping in two dimensions, these methods produce $O \times V$ data blocks (also referred to as co-clusters), denoted by $\mathbf{X}_{ov}^L$ for the lower bounds and $\mathbf{X}_{ov}^U$ for the upper bounds. Each data block is represented by a prototype, which is a representative value for that block. The prototypes are represented by the matrix $\mathbf{C} = [c_{ov}]_{\substack{1\leq o\leq O \\ 1\leq v\leq V}}$, where each prototype $c_{ov}$ is an interval of the form $c_{ov} = [c_{ov}^L, c_{ov}^U]$, with $c_{ov}^L \leq c_{ov}^U$. Here, $c_{ov}^L$ and $c_{ov}^U$ denote the prototypes associated with the lower-bound block $\mathbf{X}_{ov}^L$ and the upper-bound block $\mathbf{X}_{ov}^U$, respectively.

The proposed algorithms minimize a general objective function $J$, given by Eq. (4), that measures the local heterogeneity within each data block, based on the distances between data points and their corresponding prototypes:

$$J = \sum_{o,v,i,d} (g_{io})^m (h_{dv})^n \varphi(x_{id}, c_{ov}), \quad (4)$$

where $\varphi(\cdot)$ denotes a suitable distance function specific to each method, and $m, n \in (1, \infty)$ are the fuzzifier parameters that control the degree of fuzziness of the memberships for objects and variables, respectively.

### A. Interval Fuzzy Double K-means (IFDK)

The *Interval Fuzzy Double K-means (IFDK)* is an extension of the traditional Interval Fuzzy C-Means (IFCM) [11] for the co-clustering framework. The IFCM minimizes the following objective function

$$J_{IFDK} = \sum_{o,v,i,d} (g_{io})^m (h_{dv})^n \left[(x_{id}^L - c_{ov}^L)^2 + (x_{id}^U - c_{ov}^U)^2\right] \quad (5)$$

From a random initialization of the fuzzy partitions $\mathbf{G}$ and $\mathbf{H}$, the IFDK algorithm alternates among three steps to update the prototypes $\mathbf{C}$, the sample partition $\mathbf{G}$, and the feature partition $\mathbf{H}$. The objective function $J_{IFDK}$ is minimized using the Lagrange multiplier technique, considering the constraints imposed on each component, which leads to the following Lagrangian function:

$$\mathcal{L} = J_{IFDK} - \sum_{i=1}^{I} \gamma_i \left(\sum_{o=1}^{O} g_{io} - 1\right) - \sum_{d=1}^{D} \theta_d \left(\sum_{v=1}^{V} h_{dv} - 1\right), \quad (6)$$

where $\gamma_i$, and $\theta_d$ are the Lagrange multipliers.

The minimizers of $J_{\text{IFDK}}$ are obtained by setting the partial derivatives of $\mathcal{L}$ (Eq. (6)) with respect to each component to zero, while keeping the others fixed. By doing so, we obtain:
*Step 1. Representation:* The prototype matrix $\mathbf{C}$ has its components computed as follows:

$$c_{ov}^{L} = \frac{\sum_{i,d}(g_{io})^m (h_{dv})^n x_{id}^{L}}{\sum_{i,d}(g_{io})^m (h_{dv})^n} \tag{7}$$

$$c_{ov}^{U} = \frac{\sum_{i,d}(g_{io})^m (h_{dv})^n x_{id}^{U}}{\sum_{i,d}(g_{io})^m (h_{dv})^n} \tag{8}$$

Since $x_{id}^{L} \leq x_{id}^{U}$, by Eqs. (7) and (8), we easily conclude that $c_{ov}^{L} \leq c_{ov}^{U}$, satisfying the condition for interval data.
*Step 2. Sample assignment:* The fuzzy membership matrix of the samples $\mathbf{G}$ is updated according to the following rule:

$$g_{io} = \left[\sum_{r=1}^{O}\left(\frac{\mathcal{D}_{io}}{\mathcal{D}_{ir}}\right)^{\frac{1}{m-1}}\right]^{-1}, \tag{9}$$

where

$$\mathcal{D}_{io} = \sum_{v,d}(h_{dv})^n \left[(x_{id}^{L} - c_{ov}^{L})^2 + (x_{id}^{U} - c_{ov}^{U})^2\right] \tag{10}$$

*Step 3. Feature assignment:* The fuzzy membership matrix of the features $\mathbf{H}$ is updated according to the following rule:

$$h_{dv} = \left[\sum_{s=1}^{V}\left(\frac{\mathcal{D}'_{dv}}{\mathcal{D}'_{ds}}\right)^{\frac{1}{n-1}}\right]^{-1}, \tag{11}$$

where

$$\mathcal{D}'_{dv} = \sum_{o,i}(g_{io})^m \left[(x_{id}^{L} - c_{ov}^{L})^2 + (x_{id}^{U} - c_{ov}^{U})^2\right] \tag{12}$$

These steps are repeated until the change in the value of $J_{IFDK}$ becomes sufficiently small, indicating convergence.

### B. Interval Kernel Fuzzy Double K-means (IKFDK)

This section introduces the Interval Kernel Fuzzy Double K-means (IKFDK) algorithms. These methods are useful for grouping samples and features with nonlinear structures. We propose two variants corresponding to each kernel-based clustering approach.

*1) Interval Kernel Fuzzy Double K-means in the Original Space (IKFDK-O):* The IKFDK-O variant is an extension of the traditional Kernel Fuzzy C-Means for Interval Data (KFCM-IV) [12] to the co-clustering framework. This method defines the prototypes in the original input space and minimizes the following objective function:

$$\begin{aligned} J_{IKFDK-O} &= \sum_{o,v,i,d}(g_{io})^m (h_{dv})^n \|\Omega(x_{id}) - c_{ov}\|^2 \\ &= 2\sum_{o,v,i,d}(g_{io})^m (h_{dv})^n [1 - \mathcal{K}(x_{id}, c_{ov})], \end{aligned} \tag{13}$$

where $m$ and $n$ are defined as before, and $\mathcal{K}(\cdot)$ in given in Eq. (2).

The minimization process is similar to that of the IFDK algorithm presented previously. The minimizers of $J_{IKFDK-O}$ are obtained using the Lagrange multiplier technique. The corresponding Lagrangian function is identical to Eq. (6), with $J_{IFDK}$ replaced by $J_{IKFDK-O}$ (see Eq. (13)). Thus, the minimizers of $J_{IKFDK-O}$ are given as follows.
*Step 1. Representation:* The prototype matrix $\mathbf{C}$ has its components computed as follows:

$$c_{ov}^{L} = \frac{\sum_{i,d}(g_{io})^m (h_{dv})^n \mathcal{K}(x_{id}, c_{ov}) x_{id}^{L}}{\sum_{i,d}(g_{io})^m (h_{dv})^n \mathcal{K}(x_{id}, c_{ov})} \tag{14}$$

$$c_{ov}^{U} = \frac{\sum_{i,d}(g_{io})^m (h_{dv})^n \mathcal{K}(x_{id}, c_{ov}) x_{id}^{U}}{\sum_{i,d}(g_{io})^m (h_{dv})^n \mathcal{K}(x_{id}, c_{ov})} \tag{15}$$

Since $x_{id}^{L} \leq x_{id}^{U}$, by Eqs. (14) and (15), we easily conclude that $c_{ov}^{L} \leq c_{ov}^{U}$, satisfying the condition for interval data.
*Step 2. Sample assignment:* The fuzzy membership matrix of the samples $\mathbf{G}$ is updated according to Eq. (9), using

$$\mathcal{D}_{io} = \sum_{v,d}(h_{dv})^n \left[1 - \mathcal{K}(x_{id}, c_{ov})\right] \tag{16}$$

*Step 3. Feature assignment:* The fuzzy membership matrix of the features $\mathbf{H}$ is updated according to Eq. (11), using

$$\mathcal{D}'_{dv} = \sum_{o,i}(g_{io})^m \left[1 - \mathcal{K}(x_{id}, c_{ov})\right] \tag{17}$$

These steps are repeated until the change in the value of $J_{IKFDK-O}$ becomes sufficiently small, indicating convergence.

*2) Interval Kernel Fuzzy Double K-means in the Kernel Space (IKFDK-K):* The IKFDK-K variant assumes that the prototypes are located in the original space, here represented by $c_{ov}^{\Omega}\ \forall o, v$. This variant minimizes the following objective function:

$$J_{IKFDK-K} = \sum_{o,v,i,d}(g_{io})^m (h_{dv})^n \ \|\Omega(x_{id}) - c_{ov}^{\Omega}\|^2 \tag{18}$$

The corresponding Lagrangian function is identical to Eq. (6), with $J_{IFDK}$ replaced by $J_{IKFDK-K}$ (see Eq. (18)). Thus, we have that the minimizers of $J_{IKFDK-K}$ are given as follows.
*Step 1. Representation:* Taking the partial derivative of the Lagrangian function w.r.t $c_{ov}^{\Omega}$, we obtain:

$$c_{ov}^{\Omega} = \frac{\sum_{i,d}(g_{io})^m (h_{dv})^n \Omega(x_{id})}{\sum_{i,d}(g_{io})^m (h_{dv})^n}. \tag{19}$$

Since the mapping $\Omega$ is not explicitly known, the prototypes in the kernel space given by Eq. (19) cannot be computed explicitly. However, by substituting Eq. (19) into Eq. (1), the distances in the kernel space can be computed as follows:

$$\begin{aligned} \|\Omega(x_{id}) - c_{ov}^{\Omega}\|^2 &= 1 - 2\frac{\sum_{r=1}^{I}\sum_{s=1}^{D} g_{ro}^m h_{sv}^n \mathcal{K}(x_{id}, x_{rs})}{\left(\sum_{r=1}^{I} g_{ro}^m\right)\left(\sum_{s=1}^{D} h_{sv}^n\right)} \\ &+ \frac{\sum_{a=1}^{I}\sum_{b=1}^{D}\sum_{r=1}^{I}\sum_{s=1}^{D} g_{ao}^m h_{bv}^n g_{ro}^m h_{sv}^n \mathcal{K}(x_{ab}, x_{rs})}{\left(\sum_{r=1}^{I} g_{ro}^m\right)^2\left(\sum_{s=1}^{D} h_{sv}^n\right)^2} \end{aligned} \tag{20}$$

Equation (20) computes distances in the kernel space directly from the input space using the kernel trick, without explicitly evaluating the mapping $\Omega(\cdot)$. The second term measures the similarity between the data point and the corresponding data block, while the third term represents the overall similarity among the points within this block. Note that the third term is constant with respect to the data point and depends only on the data block. Since the second term has a negative sign, lower similarity between the data point and the data block leads to a larger distance.

*Step 2. Sample assignment:* The fuzzy membership matrix of the samples $\mathbf{G}$ is updated according to Eq. (9), using

$$\mathcal{D}_{io} = \sum_{v,d} (h_{dv})^n \|\Omega(x_{id}) - c_{ov}^{\Omega}\|^2 \tag{21}$$

*Step 3. Feature assignment:* The fuzzy membership matrix of the features $\mathbf{H}$ is updated according to Eq. (11), using

$$\mathcal{D}'_{dv} = \sum_{o,i} (g_{io})^m \|\Omega(x_{id}) - c_{ov}^{\Omega}\|^2 \tag{22}$$

These steps are repeated until the change in the value of $J_{IKFDK-K}$ becomes sufficiently small, indicating convergence.

### C. Complexity and Convergence

Assuming that the algorithms require $M$ iterations until convergence, the time complexity of the IFDK and IKFDK-O algorithms is $\mathcal{O}(MOVID)$. In contrast, the IKFDK-K variant has a time complexity of $\mathcal{O}(MOVI^2D^2)$, which is considerably higher. The main computational bottleneck lies in computing distances in the kernel space (Eq. (20)), which has a complexity of $\mathcal{O}(OVI^2D^2)$. It is worth noting that, since interval-valued datasets are generally small or medium in size [13], the applicability of the algorithm remains feasible. However, for large datasets, using IKFDK-K may become impractical.

The proposed algorithms follow an alternating optimization scheme, where each update step is obtained by minimizing the corresponding objective function with respect to one component while keeping the others fixed. Therefore, the objective functions are non-increasing at each iteration. Since they are bounded below, the sequence of values converges. Consequently, the algorithms converge to a stationary point.

### D. Algorithm

Algorithm 1 summarizes the steps of the proposed algorithms IFDK, IKFDK-O, and IKFDK-K.

**Algorithm 1** IFDK, IKFDK-O, and IKFDK-K

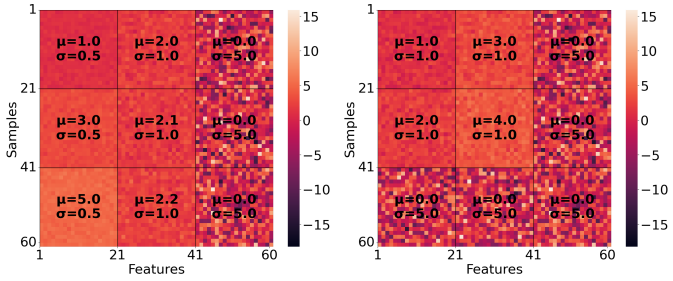
**Input:** $\mathbf{X}$, $O$, $V$, $m$, $n$, $\sigma^2$, $T$
**Initialization:** randomly initialize $\mathbf{G}$ and $\mathbf{H}$ under the required constraints. For IKFDK-O, randomly select the prototypes $\mathbf{C}$ from $\mathbf{X}$. .
**repeat**
  **Step 1: Representation** .
  IFDK: update $\mathbf{C}$ according to Eqs. (7) and (8).
  IKFDK-O: update $\mathbf{C}$ according to Eqs. (14) and (15).
  IKFDK-K: compute $|\Omega(x_{id}) - c_{ov}^{\Omega}\|^2$ based on Eq. (20).
  **Step 2: Sample assignment**
  IFDK: update $\mathbf{G}$ based on Eq. (9) using Eq. (10).
  IKFDK-O: update $\mathbf{G}$ based on Eq. (9) using Eq. (16).
  IKFDK-K: update $\mathbf{G}$ based on Eq. (9) using Eq. (21).
  **Step 3: Feature assignment**
  IFDK: update $\mathbf{H}$ based on Eq. (11) using Eq. (12).
  IKFDK-O: update $\mathbf{H}$ based on Eq. (11) using Eq. (17).
  IKFDK-K: update $\mathbf{H}$ based on Eq. (11) using Eq. (22).
**until** the change in $J_{IFDK}$, $J_{IKFDK-O}$, and $J_{IKFDK-K}$ is small or there is no change.
**Output:** $\mathbf{G}$, $\mathbf{H}$, and $\mathbf{C}$ (for IFDK and IKFDK-O).

## IV. Experimental evaluation

In this section, we evaluate the effectiveness of the proposed methods through experiments conducted on synthetic and benchmark interval-valued datasets. For comparison purposes, we consider two traditional clustering methods closely related to the proposed algorithms, namely Interval Fuzzy C-Means (IFCM) [11] and Kernel Fuzzy C-Means for Interval Data (KFCM-IV) [12].

### A. Synthetic datasets

We consider two synthetic datasets to evaluate the efficiency of the proposed methods under controlled scenarios. Each dataset consists of 60 samples and 60 features, partitioned into three clusters of 20 elements each, yielding nine data blocks of 400 points each. To generate the interval-valued datasets, we initially created seeds $\alpha_{id}$ from a Gaussian distribution with block-specific parameters, i.e., $\alpha_{id} \sim \mathcal{N}(\mu_{ov}, \sigma_{ov})$. The intervals were then constructed as $[\alpha_{id} - \beta_{id},\ \alpha_{id} + \beta_{id}]$, where $\beta_{id}$ follows a uniform distribution on the interval $[0, 2]$, i.e., $\beta_{id} \sim \mathcal{U}(0, 2)$. The parameters and a heatmap of the generated seeds for each dataset are displayed in Figure 1.




(a) Synthetic dataset 1 (b) Synthetic dataset 2

Fig. 1: The parameters ($\mu$, $\sigma$) and a heatmap of the generated seeds $\alpha_{id}$ for the synthetic datasets.

We conducted a Monte Carlo simulation with 100 replicates. In each replicate, the methods were run 50 times with different initializations, and the results corresponding to the initialization that yielded the lowest value of the objective function were used. To assess the performance of the methods, we employed the Adjusted Rand Index (ARI) [14] to quantify the agreement between an *a priori* hard partition and the hard partition produced by the algorithms. The ARI ranges from $-1$ to 1, with 1 indicating perfect agreement and values close

to 0 or negative indicating agreement due to chance. Since the feature labels are also known, we additionally employed the Co-clustering Adjusted Rand Index (CARI) [15]. The CARI is an extension of the ARI designed to compare two pairs of partitions for samples and features simultaneously, making it suitable for evaluating block structures. Hard partitions were obtained by assigning samples and features to the clusters with the highest membership values. Furthermore, in order to obtain a partition of the features for computing the CARI, we applied IFCM and KFCM-IV to group the features, since these methods do not naturally provide feature partitions.

We set the number of sample and feature clusters to three ($O = V = 3$), corresponding to the number of *a priori* classes. Furthermore, the parameters $m$ and $n$ were fixed at 1.1, and the width parameter $\sigma^2$ for the kernel-based methods was estimated according to Ref. [16] as $\sigma^2 = \frac{Q_{10}+Q_{90}}{4}$, where $Q_{10}$ and $Q_{90}$ denote the 0.1 and 0.9 quantiles, respectively, of $\|\mathbf{x}_r^L - \mathbf{x}_s^L\|^2 + \|\mathbf{x}_r^U - \mathbf{x}_s^U\|^2$ for $r \neq s$ in the traditional clustering, or $(x_{ab}^L - x_{rs}^L)^2 + (x_{ab}^U - x_{rs}^U)^2$ for $a \neq r$ and $b \neq s$ in the co-clustering framework. The maximum number of iterations was set to 100.

Table I presents the results of the experiments conducted on the synthetic datasets. In Synthetic Dataset 1 (see Figure 1(a)), the sample clusters are well separated in the first feature cluster, characterized by distinct means and low variability. In the second feature cluster, the sample clusters overlap, while the third feature cluster (20 features) corresponds to noise with respect to the sample clusters, with equal means and high variance, which makes the task challenging for traditional clustering algorithms. In this scenario, the proposed co-clustering algorithms outperform the traditional clustering techniques in terms of sample partitioning. Although KFCM-IV achieves better results than IFDK and IKFDK-O for the feature partition, the analysis of the block structure using the CARI reveals that the proposed methods yield superior performance, with IKFDK-K achieving a perfect result.

In Synthetic Dataset 2 (see Figure 1(b)), the last sample cluster represents noise for the feature clusters, while the last feature cluster represents noise for the sample clusters, as both exhibit equal means and high variance. As a result, only four distinct blocks are present. In this scenario, the proposed algorithms outperform the traditional clustering methods in all cases, demonstrating a strong ability to recover the true block structure in the presence of noise in both dimensions. In particular, the IKFDK-K variant exhibits excellent performance.

TABLE I: Mean (standard deviation) across 100 Monte Carlo replicates. $\text{ARI}_S$ and $\text{ARI}_F$ represent the ARI for samples and features, respectively. **Blue** and **red** values indicate the best and second-best results, respectively.

| Algorithm | Synthetic Dataset 1 | | | Synthetic Dataset 2 | | |
|---|---|---|---|---|---|---|
| | $\text{ARI}_S$ | $\text{ARI}_F$ | CARI | $\text{ARI}_S$ | $\text{ARI}_F$ | CARI |
| IFCM | 0.38 (0.07) | 0.44 (0.11) | 0.31 (0.06) | 0.35 (0.07) | 0.52 (0.11) | 0.33 (0.07) |
| KFCM-IV | 0.38 (0.08) | **0.88 (0.06)** | 0.50 (0.06) | 0.35 (0.08) | 0.52 (0.11) | 0.33 (0.06) |
| IFDK | 0.73 (0.18) | 0.58 (0.14) | 0.57 (0.17) | 0.60 (0.09) | **0.72 (0.11)** | 0.56 (0.09) |
| IKFDK-O | **0.95 (0.13)** | 0.66 (0.23) | **0.74 (0.19)** | **0.77 (0.12)** | 0.69 (0.14) | **0.65 (0.10)** |
| IKFDK-K | **1.00 (0.00)** | **1.00 (0.00)** | **1.00 (0.00)** | **0.94 (0.14)** | **1.00 (0.01)** | **0.96 (0.10)** |

TABLE II: Summary of the benchmark datasets.

| ID | Dataset | $I$ | $D$ | $\mathcal{C}$ |
|---|---|---|---|---|
| ABAL | Abalone | 24 | 7 | 3 |
| CARS | Car models | 33 | 8 | 4 |
| HORS | Horse | 12 | 7 | 4 |
| FISH | Freshwater fish species | 12 | 13 | 4 |
| FUNG | Fungi | 55 | 5 | 3 |
| CITY | City temperature | 37 | 12 | 4 |

The datasets can be obtained in Ref [13].

### B. Benchmark datasets

In this subsection, the algorithms are applied to six benchmark interval-valued datasets. Table II presents a summary of these datasets.

In this case, there is no *a priori* partition for the features. Therefore, our goal is to evaluate the ability of the methods to recover the true sample cluster structure. In addition to the ARI index, we also consider the Hullemeier index (HUL) [17]. The HUL also measures the agreement between two partitions, but it takes into account the fuzzy partition produced by the algorithm rather than a hard one, with the aim of capturing data overlap that is lost when hard partitions are used. The HUL lies in $[0, 1]$, with a value of 1 indicating perfect agreement between the partitions.

We set the number of sample clusters equal to the number of *a priori* classes, i.e., $O = \mathcal{C}$. For the sake of simplicity, the number of feature clusters was set equal to the number of sample clusters ($V = O$), as commonly adopted in co-clustering applications [5]–[7]. Furthermore, the parameters $m$ and $n$ were set to 1.1, and the width parameter $\sigma^2$ for the kernel-based methods was estimated according to Ref. [16], as described in Section IV-A. The algorithms were executed 100 times with different initializations, and the evaluation metrics were computed for the run that yielded the lowest value of the objective function (best fit) as well as the average value across all runs.

The results are reported in Table III. It can be observed that one of the proposed kernel-based variants achieved the best performance in most cases. Conversely, the IFDK variant did not achieve competitive results when compared to the other methods. When analyzing the average performance across all datasets (avr.), the proposed variants IKFDK-K and IKFDK-O achieved the first and second best performances, respectively, for the best solution analysis. Regarding the average solution, IKFDK-O obtained the best performance in terms of ARI, while IKFDK-K and KFCM-IV outperformed the other methods with respect to the HUL index.

In terms of average ranking (avr. rank), which represents the mean ranking position (with 1 corresponding to the best and 5 to the worst performance) of the methods across datasets, IKFDK-O and KFCM-IV achieved the best results for ARI, whereas IKFDK-K obtained the best performance for HUL, followed by IKFDK-O and IFCM, considering the best solution case. With respect to the average results, the proposed IKFDK-O variant achieved the best performance for both evaluation metrics. The KFCM-IV method presented the

TABLE III: Results on benchmark datasets. **Blue** and **red** values indicate the best and second-best performances, respectively. The value preceding the parentheses corresponds to the best result according to the objective function (lowest value), while the value in parentheses represents the average result across all initializations.

| Datasets | ARI ↑ | | | | | HUL ↑ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | IFCM | KFCM-IV | IFDK | IKFDK-O | IKFDK-K | IFCM | KFCM-IV | IFDK | IKFDK-O | IKFDK-K |
| CARS | 0.279 (0.338) | 0.279 (0.339) | 0.279 (0.293) | **0.365** (**0.391**) | **0.354** (**0.355**) | 0.661 (0.740) | 0.661 (**0.742**) | 0.661 (0.682) | **0.722** (0.715) | **0.763** (**0.763**) |
| FUNG | **0.396** (**0.232**) | 0.339 (**0.281**) | 0.022 (0.036) | 0.311 (0.171) | **0.422** (0.093) | **0.726** (**0.637**) | 0.696 (**0.668**) | 0.521 (0.529) | 0.677 (0.609) | **0.747** (0.574) |
| HORS | **0.366** (0.303) | **0.366** (0.305) | **0.366** (**0.362**) | **0.427** (**0.390**) | **0.366** (0.354) | **0.788** (0.759) | **0.788** (0.761) | **0.788** (**0.789**) | **0.803** (**0.798**) | **0.788** (0.784) |
| CITY | **0.566** (**0.537**) | **0.566** (0.523) | 0.546 (0.534) | 0.546 (**0.536**) | 0.546 (0.503) | **0.793** (**0.779**) | **0.793** (0.774) | 0.785 (**0.779**) | 0.785 (**0.780**) | 0.789 (0.770) |
| FISH | **-0.047** (-0.030) | **-0.047** (-0.024) | **-0.047** (-0.028) | **0.103** (**0.055**) | -0.061 (**0.084**) | 0.515 (0.611) | 0.515 (0.629) | 0.515 (0.588) | **0.663** (**0.651**) | **0.606** (**0.683**) |
| ABAL | **0.020** (**0.053**) | **0.020** (**0.053**) | **0.020** (0.047) | **0.020** (**0.103**) | **0.193** (0.039) | 0.580 (0.585) | 0.580 (0.585) | **0.582** (**0.588**) | 0.581 (**0.594**) | **0.608** (0.585) |
| avr. ↑ | 0.263 (0.239) | 0.254 (**0.246**) | 0.198 (0.207) | **0.295** (**0.274**) | **0.303** (0.238) | 0.677 (0.685) | 0.672 (**0.693**) | 0.642 (0.659) | **0.705** (0.691) | **0.717** (**0.693**) |
| avr. rank ↓ | **2.000** (3.167) | 2.167 (**2.833**) | 2.833 (3.833) | **2.000** (**1.667**) | 2.333 (3.333) | **2.500** (3.167) | 2.667 (**2.833**) | 3.167 (3.500) | **2.500** (**2.000**) | **1.667** (**2.833**) |

second-best performance, tied with the IKFDK-K variant in terms of HUL.

The IFDK variant did not achieve competitive results when compared to the other methods, exhibiting the overall worst performance. Since this method is based on a squared Euclidean distance measure, the presence of outliers in the datasets may have affected the model, whereas such effects are less expressive in kernel-based methods.

## V. Conclusions

In this paper, we introduced the first co-clustering algorithms for interval-valued data. We proposed the *Interval Fuzzy Double K-means (IFDK)* algorithm, as well as its kernelized extensions, namely the *Interval Kernel Fuzzy Double K-means in the Original Space (IKFDK-O)* and the *Interval Kernel Fuzzy Double K-means in the Kernel Space (IKFDK-K)*, which employ the Gaussian kernel to identify nonlinear cluster structures. Experiments conducted on synthetic and benchmark interval-valued datasets demonstrated the effectiveness of the proposed methods compared to traditional interval fuzzy clustering approaches, particularly in scenarios involving noisy data, as illustrated by the synthetic datasets. On the benchmark datasets, the proposed kernel-based algorithms achieved superior or competitive results, whereas the IFDK method was outperformed by the other methods.

For future work, we plan to investigate the impact of the parameters $m$ and $n$ on the performance of the models and to explore the incorporation of other kernel functions, such as the polynomial kernel. Finally, we also intend to study strategies to reduce the computational complexity of the IKFDK-K variant, which exhibited the best overall performance but suffers from high computational cost.

## Acknowledgements

The authors would like to thank the anonymous referees for their careful revision, the Conselho Nacional de Desenvolvimento Científico e Tecnológico-CNPq (311164/2020-0), and the Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco-FACEPE (IBPG-1424-1.03/22), for their partial financial support of this study.

## References

[1] H.-H. Bock and E. Diday, *Analysis of symbolic data: exploratory methods for extracting statistical information from complex data*. Springer Science & Business Media, 2012.

[2] A. Iftikhar, M. A. Ghazanfar, M. Ayub, S. A. Alahmari, N. Qazi, and J. Wall, "A reinforcement learning recommender system using bi-clustering and markov decision process," *Expert Systems with Applications*, vol. 237, p. 121541, 2024.

[3] C. Fettal, L. Labiod, and M. Nadif, "Boosting subspace co-clustering via bilateral graph convolution," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[4] Z. M. Husseini, M. H. F. Zarandi, and A. Ahmadi, "Type2 soft biclustering framework for alzheimer microarray," *Applied Soft Computing*, vol. 152, p. 111227, 2024.

[5] J. N. A. de Sá, M. R. Ferreira, and F. d. A. de Carvalho, "Kernel-based fuzzy co-clustering in feature space with automated variable weighting," in *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2022, pp. 1–8.

[6] J. N. A. de Sá, M. R. P. Ferreira, and F. A. T. de Carvalho, "Fuzzy and crisp gaussian kernel-based co-clustering with automatic width computation," *IEEE Transactions on Fuzzy Systems*, vol. 33, no. 6, pp. 1977–1991, 2025.

[7] J. N. A. de Sá, M. R. P. Ferreira, and F. A. T. de Carvalho, "A dual-weighted gaussian kernel-based method for fuzzy co-clustering," in *2025 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2025, pp. 1–6.

[8] B. J Mercer, "Functions of positive and negative type, and their connection the theory of integral equations," *Phil. Trans. R. Soc. Lond. A*, vol. 209, no. 441-458, pp. 415–446, 1909.

[9] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE transactions on neural networks*, vol. 12, no. 2, pp. 181–201, 2001.

[10] N. R. Pal and K. Sarkar, "What and when can we gain from the kernel versions of c-means algorithm?" *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 2, pp. 363–379, 2013.

[11] F. d. A. de Carvalho, "Fuzzy c-means clustering methods for symbolic interval data," *Pattern Recognition Letters*, vol. 28, no. 4, pp. 423–437, 2007.

[12] B. A. Pimentel, A. F. da Costa, and R. M. de Souza, "Kernel-based fuzzy clustering of interval data," in *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*. IEEE, 2011, pp. 497–501.

[13] J. N. A. de Sá, M. R. P. Ferreira, and F. d. A. T. de Carvalho, "Kernel clustering with automatic variable weighting for interval data," *Neurocomputing*, vol. 650, p. 130849, 2025.

[14] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[15] V. Robert, Y. Vasseur, and V. Brault, "Comparing high-dimensional partitions with the co-clustering adjusted rand index," *Journal of Classification*, vol. 38, no. 1, pp. 158–186, 2021.

[16] B. Caputo, K. Sim, F. Furesjo, and A. Smola, "Appearance-based object recognition using svms: which kernel should i use?" in *Proc of NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision, Whistler*, vol. 2002, 2002.

[17] E. Hullermeier, M. Rifqi, S. Henzgen, and R. Senge, "Comparing fuzzy partitions: A generalization of the rand index and related measures," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 3, pp. 546–556, 2011.