Introduction to Machine Learning: Work 3

Pedro Agúndez[*], Bruno Sánchez[*], María del Carmen Ramírez[*], Antonio Lobo[*]

December 8, 2024

Abstract

Abstract

[*] Universitat de Barcelona
pedro.agundez@estudiantat.upc.edu
bruno.sanchez.gomez@estudiantat.upc.edu
maria.del.carmen.ramirez@estudiantat.upc.edu
antonio.lobo@estudiantat.upc.edu

1 Introduction

Introduction.

2 Methodology

Introduction to methodology.

2.1 K-Means

K-Means intro.

2.1.1 Hyperparameters

1. **k:**
    - The number of clusters to partition the dataset. Determines the complexity and granularity of the clustering.
2. **Distance Metrics:**
    - **Euclidean Distance:** Calculates the root of the sum of squared differences between feature values. Standard metric for continuous data:

      $$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

    - **Manhattan Distance:** Computes the sum of absolute differences between feature values, suitable for both categorical and continuous data:

      $$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

    - **Clark Distance:** Accounts for proportional differences between feature values, enhancing interpretability for attributes with varying scales:

      $$d(x, y) = \sqrt{\sum_{i=1}^{n} \left(\frac{|x_i - y_i|}{x_i + y_i + \epsilon} \right)^2}$$

      where ϵ is a small constant to avoid division by zero.
3. **Additional Parameters:**
    - **Initial Centroids:** Pre-defined initial cluster centers used as the starting point for the clustering algorithm.
    - **Maximum Iterations:** Limits the number of iterations to prevent excessive computational time, with a default of 10 iterations.

2.1.2 Clustering Methodology

- **Clustering Process:**
    1. Assign each data point to the nearest centroid using the specified distance metric.
    2. Recalculate centroids by computing the mean of all points in each cluster.
    3. Repeat assignment and recalculation until convergence or maximum iterations are reached.
- **Convergence Criteria:**

  - Clusters are considered stable when centroids no longer significantly change between iterations.

- **Variance Computation:**
  - Total within-cluster variance (E) is calculated by summing squared distances of points to their respective cluster centroids.
  - Provides a measure of clustering compactness and quality.

This methodology allows for flexible clustering configurations, enabling analysis across different datasets and hyperparameter values.

2.2 Global K-Means

Out of the proposed improvements to the K-Means algorithm, the first we chose to implement was the **Global K-Means** algorithm [3], which focuses on following a deterministic and systematic approach to "optimal" centroid initialization and cluster formation. Additionally, we have also implemented the improvements to the Global K-Means algorithm itself, proposed in the original article by Likas et al.: *Fast Global K-Means*, and *Initialization with k-d Trees.* By addressing the limitations of traditional K-Means, this enhanced methodology introduces novel strategies, including PCA-based data partitioning and iterative error-reduction mechanisms, to improve both accuracy and computational efficiency.

This section outlines the hyperparameter configurations and clustering methodology adopted for the Global K-Means algorithm, which was implemented in the `GlobalKMeansAlgorithm` class.

2.2.1 Hyperparameters

We consider the same hyperparameters as for the standard K-Means algorithm (Section 2.1), except for 2 significant modifications:

1. **Initial Centroids:**
   - Global K-Means no longer accepts a collection of initial centroids as a hyperparameter, since the goal of this algorithm is rooted in the deterministic calculation of the "best possible" centroids, which substitutes their random initialization.
2. **Number of Buckets:**
   - Controls initial data partitioning, by defining the number of candidate points that we will consider as possible centroids throughout the algorithm.
   - Its default value is $2 \cdot k$, but we also test values $3 \cdot k$ and $4 \cdot k$.

2.2.2 Clustering Methodology

- **Initialization with k-d Trees:**
  1. Use k-d tree partitioning based on Principal Component Analysis (PCA).
  2. Recursively partition data samples into buckets.
  3. Select candidate points based on bucket centroids.
- **Fast Global K-Means Algorithm:**
  1. Initialize first centroid as dataset mean.
  2. Iteratively add centroids by:
     - For each $k' = 2, \ldots, k$, we already have $k'-1$ centroids.
     - Compute guaranteed error reduction for candidate points with respect to the $k'-1$ centroids,

$$b_n = \sum_{j=1}^{N} \max\left(d_{k'-1}^{j} - ||x_n - x_j||^2, 0\right) \ ,$$

where $d_{k'-1}^{j}$ is the squared distance between x_j and the closest centroid among the $k'-1$ obtained so far. The pair-wise squared distances between points are precomputed at the start.

- Select point with maximum guaranteed error reduction.
  - Run k'-means with the $k'-1$ centroids plus the selected point, unitl convergence.

3. Repeat until k clusters are formed.

This methodology provides a sophisticated approach to centroid initialization and clustering, leveraging PCA-based partitioning and error reduction strategies in order to achieve an improvement in consistency and speed with respect to the base K-Means algorithm.

2.3 Fuzzy C-Means

We selected the **generalized suppressed Fuzzy C-Means** (gs-FCM) algorithm, an improvement over traditional FCM, which often shows multimodal behavior near cluster boundaries (Fig. 1a). This issue, where fuzzy memberships remain high for unrelated clusters, was addressed by Höppner and Klawonn [2].

The suppressed Fuzzy C-Means (s-FCM) algorithm [1] enhances convergence speed and classification accuracy without minimizing the traditional objective function J_{FCM}. It introduces a suppression step during each iteration to reduce non-winner fuzzy memberships, which is mathematically equivalent to virtually reducing the distance to the winning cluster's prototype (Fig. 1b) [5].

Szilágyi et al. [5] defined the quasi-learning rate η of s-FCM, analogous to learning rates in competitive algorithms:

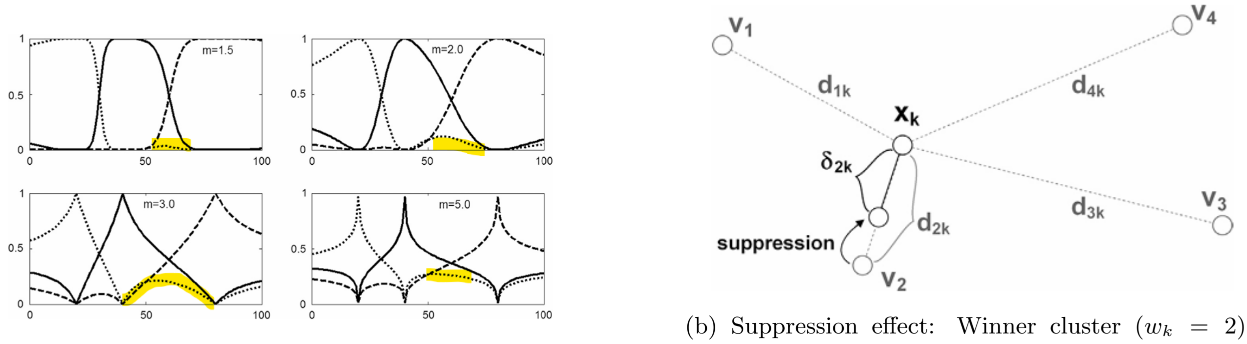
$$\eta(m, \alpha, u_{wk}) = 1 - \frac{\delta_{wk}}{d_{wk}} = 1 - \left(1 + \frac{1-\alpha}{\alpha u_{wk}}\right)^{(1-m)/2}.$$

Building on this, gs-FCM modifies the learning rate to decay linearly with increasing winner membership u_{wk} for faster convergence, as proposed in sg_{ρ}-FCM [4]:

$$\eta(u_{wk}) = 1 - \rho u_{wk}, \quad \text{where } 0 \leq \rho \leq 1.$$

This approach ensures a logical adaptation of membership weighting, expressed as:

$$\alpha_k = \left[1 - u_w + u_w \left(1 - f(u_w)\right)^{2/(1-m)}\right]^{-(1-m)}.$$



(a) Multimodal fuzzy memberships near cluster boundaries for varying fuzzy exponent m.

(b) Suppression effect: Winner cluster ($w_k = 2$) gains increased membership while non-winners are suppressed.

Figure 1: Figures adapted from [4].

3 Results

To systematically evaluate the different configurations of each clustering algorithm, the following procedure is followed to extract results for each of the 3 data sets, in order to later perform an analysis those results:

1. **Data Preparation**: The dataset is loaded and the data samples are separated from their labels into separate 2 sets. This way, we perform the clustering analysis in a completely non-supervised way, and we then utilize the labels to extract supervised metrics of the cultering results.

2. **Parameter Configuration**: A comprehensive set of values for the algorithm's hyperparameters is defined. These combinations reflect various ways to tune the clustering algorithm.

3. **Model Evaluation**: For each parameter combination, the clustering algorithm is applied on the unlabeled data and then evaluated with different metrics. This step yields the following metrics: Adjusted Rand Score (ARI), Normalized Mutual Information (NMI), Davies-Bouldin Index(DBI), Silhouette score, Calinski-Harabasz Score (CHS), and execution time. Together, these metrics (the first 2 supervised, and the rest non-supervised) measure the effectiveness and efficiency of the clustering.

4. **Results Compilation**: The performance metrics for each parameter combination are recorded in a structured format. These results are saved as a dataset that summarizes the outcomes of all evaluations, forming a basis for analysis. We save as well the cluster labels of all samples for each clustering algorithm that we run, so we can recover the same clusters in the posterior analysis.

5. **Results Analysis**: After results are compiled across all configurations, quantitative and qualitative analysis is performed to identify common trends among the best-performing configurations. This analysis helps determine the most reliable and effective parameter settings for accurate and efficient clustering. We will discuss our results in the following sections.

   Note: Since all of the tested datasets have high dimensionality, we use Principal Component Analysis (PCA) to extract the 2 principal components in order to generate the clustering scatter plots of the datasets for visualization purposes.

3.1 K-Means

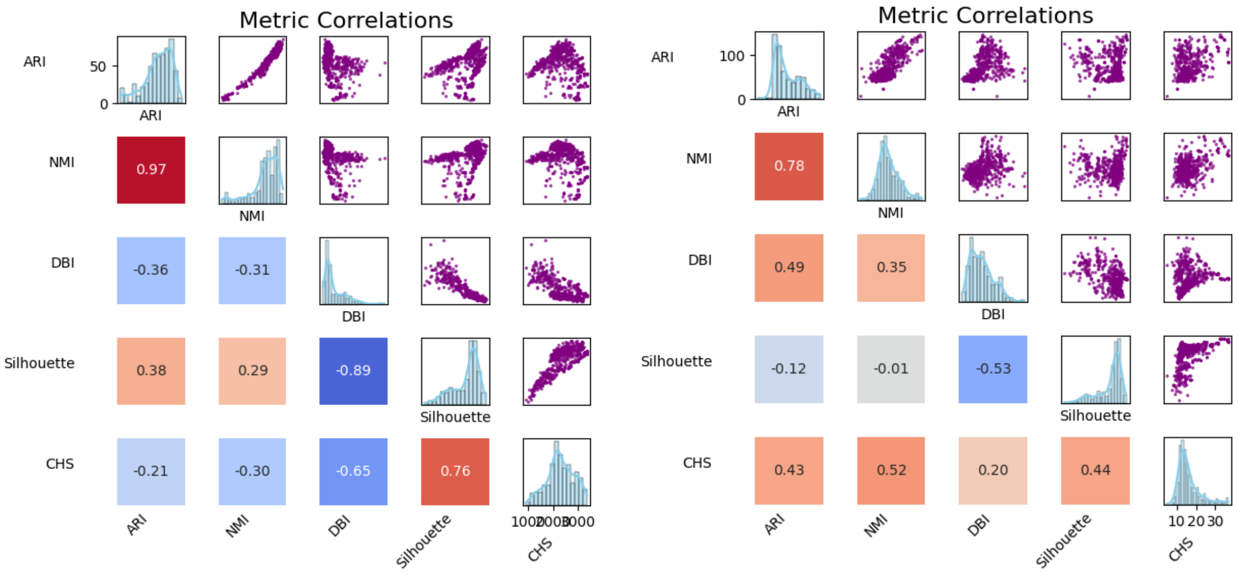
We have tested on each dataset 57 different configurations of the K-Means algorithm, by using the 3 different distance metrics with 19 different values of the k (from 2 to 20). For each of these configurations, we have run the K-Means 10 times, in order to account for the randomness of the centroid initialization. This results in a total of 570 runs of the K-Means algorithm for each dataset. From the evaluation metrics extracted for each of these runs, we study the effect of each of the 2 hyperparameters and study the best performing runs according to each of the metrics.

3.1.1 Hyperparameter Study

Before starting with the study of the best runs themselves, let us first observe some preliminary patterns about the measured metrics and the effect of each hyperparameter on the clustering performance.

In Figure 2 we summarize the relationship between the different metrics that were measured for two datasets. Each is a matrix plot where the lower triangle is a heatmap of the Pearson correlations between each pair of metrics, the diagonal elements are the histogram distributions of values of each metric, and the upper triangular has for each pair of metrics the plot of their values for all runs. It is interesting to observe that, while we would expect all of the metrics to agree on the identified trends, there are some cases where the opposite behavior is displayed.(the plots can be found in the `code` floder).

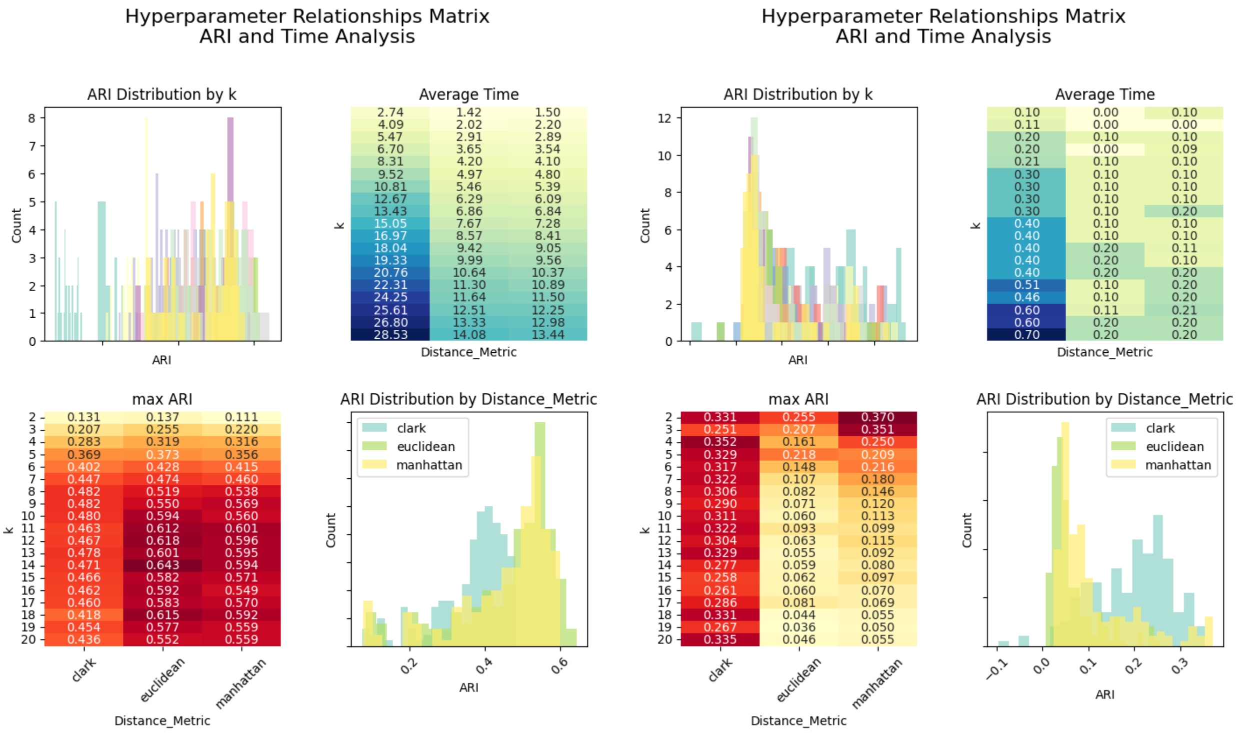
Parallelly, a different set of interesting relationship are displayed in Figure 3, where we can see heatmaps of the ARI and the Time across the different pairwise hyperparameter configurations. We can observe a general trend regarding execution time: it seems to have considerably larger values for the Clark distance metric compared to those of the other 2, which reflects the higher computational cost that this distance metric has. Additionally, we see a noteworthy divergence in the ARI trends: in the Pen-based dataset (which has 10 classes), bigger values of k (greater than 7) seem to achieve the best scores, with a somewhat lower performance with the Clark distance than with the other two; meanwhile, in the Hepatitis dataset (which has 2 classes), lower values of k seem to achieve a better ARI, with a significantly better performance with the Clark distance metric than with the other two. The behavior with respect to the k was to be expected due to the true number of labels of each dataset, yet it still is compelling to see it reflected so clearly in the results. On the other hand, it is enlightening to see opposite behavior with respect to the distance metric, which reflects that the Clark distance fails to capture some properties of the Pen-based dataset, while it excells to do so within the Hepatitis dataset.



(a) Pen-based metric correlations

(b) Hepatitis metric correlations

Figure 2: Metric correlations and distributions in two datasets



(a) Pen-based pairplot matrix

(b) Hepatitis pairplot matrix

Figure 3: Hyperparameter pairplot matrices based on F1 Score and Time

3.1.2 Best Runs

4 Conclsuion

Conclusion.

References

[1] J.L. Fan, W.Z. Zhen, and W.X. Xie. Suppressed fuzzy c-means clustering algorithm. Pattern Recognition Letters, 24:1607–1612, 2003.

[2] F. Höppner and F. Klawonn. Improved fuzzy partitions for fuzzy regression models. International Journal of Approximate Reasoning, 32:85–102, 2003.

[3] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. Pattern Recognition, 36(2):451–461, 2003.

[4] László Szilágyi and Sándor M. Szilágyi. Generalization rules for the suppressed fuzzy c-means clustering algorithm. Neurocomputing, 139:298–309, 2014.

[5] L. Szilágyi, S.M. Szilágyi, and Z. Benyó. Analytical and numerical evaluation of the suppressed fuzzy c-means algorithm: a study on the competition in c-means clustering models. Soft Computing, 14:495–505, 2010.