Likelihood Optimization of Probabilistic Fuzzy C-Means based on k-NN and Local Search

Davide Cazzorla
Dept. of Computer Science
University of Bari Aldo Moro
Bari, Italy
Email: d.cazzorla@phd.uniba.it

Corrado Mencar
Dept. of Computer Science
University of Bari Aldo Moro
Bari, Italy
Email: corrado.mencar@uniba.it

***Abstract*—Fuzzy C-Means (FCM) is a widely used clustering algorithm that offers robustness, efficiency, and simplicity. However, its parameters (the membership matrix) and hyperparameter (the fuzzification coefficient) can be interpreted in probabilistic terms, thus opening the door for the application of a probabilistic methodology for clustering that is alternative to classical soft clustering methods like Gaussian Mixture Models (GMM). In particular, the Probabilistic FCM (PFCM) is inspired by FCM but adopts a different objective function that is based on negative log-likelihood (NLL). Based on the hypothesis that the optimal values of prototypes are close to data points, this paper describes an optimization algorithm using k-NN and local search that significantly reduces both the computation time and the NLL in comparison to a standard optimization based on the basin-hopping algorithm. Furthermore, a preliminary experimental comparison of PFCM with FCM and GMM shows that prototypes resulting from PFCM are always located in data-dense regions, thus providing more significant clustering of data.**

***Index Terms*—soft clustering, probabilistic models, fuzzy c-means, Gaussian mixture model**

I. Introduction

Clustering is an unsupervised machine learning task whose goal is to find hidden groups of similar instances in data. In prototype-based clustering, these groups are represented by a prototypical value, while data membership to each cluster is usually represented through a membership matrix.

In critical domains like healthcare, clustering is helpful for several tasks, e.g., in finding subtypes of diseases and predicting treatment response [1], segmenting medical images [2] and in the optimization of hospital procurement [3]. In such domains, it is of utmost importance that clusters do represent real aggregations of data points and do not suffer from averaging effects that may result in cluster prototypes that are not surrounded by a meaningful cloud of data.

One widely used clustering method is Fuzzy C-Means (FCM) [4]. FCM offers key advantages such as simplicity (few hyperparameters are required), stability w.r.t. initialization, and efficient cluster computation. Moreover, FCM allows gradual membership assignment of data to clusters, thus enabling the representation of membership uncertainty. This topic has been the subject of further analysis in relation to the constraints imposed by FCM in defining cluster membership; in particular, a probabilistic interpretation was recently established to give a grounded semantics of membership degrees as well as the fuzzification coefficient [5].

From this probabilistic interpretation, a soft clustering method was proposed that is inspired by FCM but is completely grounded in probability theory [6]. This new method, which will be referred to as Probabilistic Fuzzy C-Means (PFCM), has been compared with FCM on synthetic data, showing interesting experimental results, especially in terms of the representativeness of the resulting prototypes.

The original version of PFCM was based on the basin-hopping algorithm for optimizing its objective function, which is defined in terms of the negative log-likelihood of the data [7]. While effective, basin-hopping is inefficient because it is a general optimization algorithm that does not consider the specificity of the PFCM objective function. In this paper, a different algorithm is presented, which takes advantage of the peculiarities of the PFCM objective function to generate the cluster prototypes. Moreover, the proposed algorithm uses a k-NN approach to avoid rescanning the dataset in each iteration; this leads to a significant reduction in computation time that could be beneficial for large datasets.

In the next Sec. II, PFCM is presented starting from the original FCM method, with highlights on the objective function based on negative log-likelihood. The optimization algorithm is presented in Sec. III. In the experimental Sec. IV, the proposed algorithm is compared with basin-hopping to show the main differences in the resulting clusters, while PFCM is compared against FCM and standard Gaussian Mixture Models (GMM) soft clustering on some synthetic datasets to show their different behaviors. The concluding remarks highlight current limitations and possible developments of PFCM.

The code of the paper has been published in Zenodo and it is available at https://doi.org/10.5281/zenodo.18429472.

II. Preliminaries

A. Fuzzy C-Means (FCM)

Given a dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ with N items $\mathbf{x}_j \in \mathbb{R}^d$ to be clustered into c groups, FCM aims to minimize the following objective function:

$$J(\mathbf{U},\mathbf{V}) = \sum_{j=1}^{N}\sum_{i=1}^{c} u_{ji}^{m}||\mathbf{x}_j - \mathbf{v}_i||^2 \tag{1}$$

subject to $0 \leq u_{ji} \leq 1$ and:

$$\sum_{i=1}^{c} u_{ji} = 1 \tag{2}$$

where $\mathbf{U}$ is a $(N \times c)$ matrix containing the membership values u_{ji} of the data items, $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}$ is the set of prototypes, m is the fuzziness parameter that quantifies how much the clusters overlap, and $||*||$ is a norm that quantifies the similarity between the data and the prototypes.

To find the optimal values for $\mathbf{U}$ and $\mathbf{V}$, the most common approach is to randomly initialize the prototypes and alternatively perform the following assignments:

$$u_{ji} = \frac{1}{\sum_{k=1}^{c}\left(\frac{||\mathbf{x}_j - \mathbf{v}_i||}{||\mathbf{x}_j - \mathbf{v}_k||}\right)^{\frac{2}{m-1}}} \tag{3}$$

$$\mathbf{v}_i = \frac{\sum_{j=1}^{N} u_{ji}^{m}\mathbf{x}_j}{\sum_{j=1}^{N} u_{ji}^{m}} \tag{4}$$

The algorithm is stopped when the change of the prototypes between the iterations is small enough. This procedure converges to at least a local minimum.

B. Probabilistic Fuzzy C-Means (PFCM)

A common interpretation of the membership degree in FCM is that of "degree of sharing" [8] which is intuitively appealing but not supported by an established theory. This difficulty originates from the constraint (2) that requires an operation (summing membership degrees) that is not grounded in fuzzy set theory.

By reconsidering FCM in a probabilistic setting, membership degree can be immediately interpreted as posterior probabilities, in the same way as classical soft clustering methods but with a peculiar likelihood function [5].

In PFCM, the approach to clustering is based on probability theory. Specifically, it is assumed that the data is distributed as a mixture of symmetric Pareto-like distributions:

$$f(\mathbf{x}|\theta) = \sum_{i=1}^{c} \Pr(C=i) f(\mathbf{x}|C=i,\theta) \tag{5}$$

$$= \sum_{i=1}^{c} \Pr(C=i)\left(\frac{1}{K}||\mathbf{x} - \mathbf{v}_i||_2^{-\beta}\right) \tag{6}$$

where $\theta = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c, \beta)$ are the unknown parameters, K is a normalization constant for the component densities $f(\mathbf{x}|C=i,\theta)$, $\Pr(C=i)$ is the prior cluster probability, and β is the shape parameter that controls how much probability mass is in the tails of the distribution.

The normalization constant K is necessary to have a proper density function. (Its derivation can be found in [6].) For this distribution, it is furthermore assumed that the distances are bounded, i.e., $a \leq ||\mathbf{x} - \mathbf{v}_i|| \leq l \;\forall\; i$. This assumption is required to calculate the normalization constant.

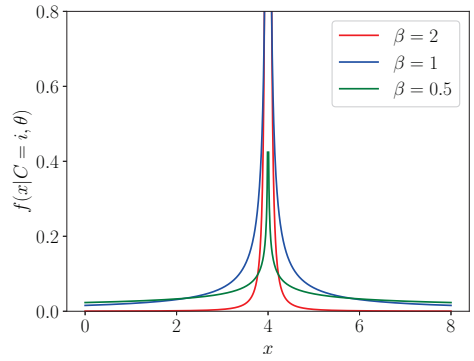


Fig. 1: Component density assumed by the PFCM for β between 0.5 and 2 with the fixed parameters $v = 4, a = 10^{-2}$ and $l = 30$.

A visual representation of the component density of one cluster can be seen in Fig. 1. A lower value of β corresponds to more probability mass in the tails. From this distribution, it can be seen that the likelihood of points away from the prototypes decreases following the power law. This characteristic is useful when the clusters in the data are blobs with high density.

Under this data distribution, and assuming $\Pr(C=i) = 1/c$, a connection between the FCM and the PFCM can be established [5]. Specifically, the membership values (3) of FCM correspond to the posterior probabilities $\Pr(C=i|\mathbf{x},\theta)$, while the β parameter is connected to the fuzziness parameter m by the formula $\beta = {}^{2}/_{m-1}$.

For the estimation of the parameters in θ, the classical approach used in statistics is to maximize the log-likelihood. However, to maintain the similarity with the objective function of the FCM, equivalently, it is minimized the negative log-likelihood (NLL):

$$-\log \mathcal{L}(\theta|\mathbf{X}) = -\log f(\mathbf{X}|\theta) \tag{7}$$

$$= -\sum_{j=1}^{N} \log \sum_{i=1}^{c} \frac{1}{c}\frac{1}{K}||\mathbf{x}_j - \mathbf{v}_i||^{-\beta} \tag{8}$$

Due to the form of the distribution (Fig. 1), the landscape of the objective function presents different valleys and holes (see Fig. 2). To mitigate this problem, in the original paper [6], the authors proposed to use the basin-hopping algorithm, which is often used with this kind of landscape.

The basin-hopping algorithm starts with an initial value for the unknown parameters and performs three steps. First, it does a random perturbation of the parameters, then it performs a local minimization, and lastly, it accepts or rejects the new parameters based on the new value reached in the minimization. In the implementation used in the original article, the local minimization was performed with the Powell method, a method that does not require calculating the derivative

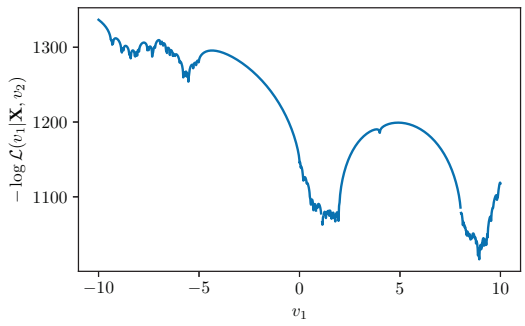


Fig. 2: Example of negative log-likelihood function for one-dimensional clustering with two prototypes. The graph depicts the objective function by varying one of the two prototypes. For the fixed prototype v_2, it was set to -6.878.

of the function. For the acceptance test, the original article employed the Metropolis criterion, which is commonly used in basin-hopping implementations, even though other tests are available.

Thanks to its stochastic nature, the basin-hopping algorithm is capable of easily escaping bad-performing local minima by "jumping" from one basin to another. However, the algorithm performs a search on the entire data space, thus resulting in inefficiency for mid-to-large data sets. Moreover, basin-hopping is a general optimization algorithm; by considering the specificity of the clustering problem, more efficient solutions can be obtained.

III. Proposed Optimization Procedure

The optimization procedure we are proposing is based on the hypothesis that the optimal prototypes could be only at a distance a from any data points, where a is the hyperparameter of eq. (6) which defines the support of the distribution.

This hypothesis is supported by the shape of the objective function (see Fig. 2 for an example) and considering that, for a sufficiently small value of a, the negative log-likelihood can take very small values around the points in the dataset. Since the objective function is the sum of negative likelihoods, then an optimal point will be close (i.e., with distance a) to a data point where the probability of observing surrounding data is maximal.

Based on this hypothesis, it is not necessary to consider the entire data space for optimization; rather, only a neighborhood is enough onto which a local search can be performed. (In the limiting case, the neighborhood may coincide with the entire dataset.)

The proposed optimization procedure is reported in Alg. 1. It starts by randomly selecting c data points and setting the initial prototypes at a distance a from them. (It suffices to add a in one coordinate only to ensure distance.) Afterward, a prototype is selected, and its k nearest neighbors are extracted. In the case that $k = N$, the k-NN is not executed, all data are considered as neighbors to reduce the number of computations. For each neighbor, a is added to the first component, and the negative log-likelihood is computed considering this point as the new position for the prototype. After calculating all the negative log-likelihoods, if there is a data point with a negative log-likelihood less than the initial prototype, the position of the prototype is updated. Subsequently, this iteration is repeated for the successive prototypes until all prototypes have been selected for the optimization. Once this loop ends, it is checked if at least one prototype was changed, and if yes, the optimization of the prototypes is started again. Otherwise, the procedure is stopped and a minimum is reached.

Algorithm 1 PFCM optimization algorithm

Require: $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}, \mathbf{x}_j \in \mathbb{R}^d$ ▷ *Dataset*
Require: $c \in \mathbb{N}, c > 1$ ▷ *Num. of clusters*
Require: $k \in \mathbb{N}, 1 \leq k \leq N$ ▷ *Num. of neighbors*
Require: $\beta > 0$
Require: $a > 0$ ▷ *< Minimum distance from data points*
Require: $l > a$ ▷ *> Maximum distance from data points*
Ensure: $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}, \mathbf{v}_i \in \mathbb{R}^d$ ▷ *Prototypes locally optimizing (8)*

▷ *Initialization* ◁
$\mathbf{a} \leftarrow [a, 0, \ldots, 0]$
for $i = 1, 2, \ldots, c$ **do**
    Randomly select a data point $\mathbf{x} \in \mathbf{X}$
    $\mathbf{v}_i \leftarrow \mathbf{x} + \mathbf{a}$
$\mathbf{V} \leftarrow \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}$
▷ *Update all prototypes* ◁
repeat
    $\mathbf{V}_{\text{old}} \leftarrow \mathbf{V}$
    ▷ *Update a single prototype* ◁
    for $i = 1, 2, \ldots, c$ **do**
        $\theta \leftarrow (\mathbf{v}_1, \ldots, \mathbf{v}_c, \beta)$
        $\min_{\text{NLL}} \leftarrow -\log \mathcal{L}(\theta|\mathbf{X})$
        $\min_{\mathbf{v}} \leftarrow \mathbf{v}_i$
        ▷ *Extract neighborhood* ◁
        if $k = N$ **then**
            $NB \leftarrow \mathbf{X}$
        else
            $NB \leftarrow$ K-NN$(\mathbf{X}, \mathbf{v}_i, k)$
        ▷ *Local search* ◁
        for all $\mathbf{x} \in NB$ **do**
            $\theta \leftarrow (\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}, \mathbf{x} + \mathbf{a}, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_c, \beta)$
            $\text{NLL} \leftarrow -\log \mathcal{L}(\theta|\mathbf{X})$
            if $\text{NLL} < \min_{\text{NLL}}$ **then**
                $\min_{\text{NLL}} \leftarrow \text{NLL}$
                $\min_{\mathbf{v}} \leftarrow \mathbf{x} + \mathbf{a}$
        ▷ *New prototype is the best neighbor* ◁
        $\mathbf{v}_i \leftarrow \min_{\mathbf{v}}$
    $\mathbf{V} \leftarrow \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}$
until $\mathbf{V} = \mathbf{V}_{\text{old}}$
▷ *Ends when no updates* ◁

The optimization algorithm can be extended to optimize the

prior probabilities $\Pr(C = i)$ and the parameter β. This can be achieved by adding specific optimization steps (e.g., based on Newton or Quasi-Newton methods) inside the main loop. However, for the sake of simplicity, these additional steps have not been considered in this work.

The computational complexity of the algorithm is $\mathcal{O}(c^2kN)$ considering k calculations of the negative log-likelihood (with complexity $\mathcal{O}(cN)$) for each of the c prototypes. The adoption of k-NN reduces the need for computing the negative log-likelihood on the entire dataset but introduces a new hyperparameter (k) that should be calibrated experimentally. For small datasets, it is possible to get rid of it by setting k to N, thus avoiding the application of the kNN procedure. The outer loop does not directly depend on data; empirically, only few loops are required before convergence.

IV. Experiments

The experiments presented will aim to test the proposed optimization algorithm for PFCM against the basin-hopping algorithm, as well as to evaluate PFCM with alternative methods on synthetic data

A. Basin-Hopping versus Proposed Optimization Method

In this experiment, the PFCM with basin-hopping was confronted with the PFCM with the proposed optimization algorithm. To assess the improvement of using k-NN, different values of k have been tested. The comparison was performed on three synthetic datasets with different numbers of clusters (namely, 2, 4, and 7 clusters, respectively).

To isolate the effect of the number of clusters on the performance, all the datasets were sampled from Gaussian distributions, and all of them consist of 200 samples. Furthermore, to allow a fair comparison, whenever possible, the methods were initialized with the same hyperparameters. In particular, for the hyperparameter β, it was set to 2 for both of the methods. Both algorithms were implemented in Python with NumPy as the numeric-processing library.

Overall, the proposed optimization algorithm is significantly faster than basin-hopping while achieving equal or lower values of the objective function (8) in most cases.

Table I reports the results in detail. It is observed that the proposed optimization algorithm surpasses basin-hopping both in terms of negative log-likelihood (NLL) and computation time for $k \geq 10$. Even in the case of using all data, the time required for convergence is significantly smaller than basin-hopping. These results confirm the hypothesis that optimal positions of prototypes are close to data points, while basin-hopping may miss good local minima when the search space becomes high.

We also observe that, as long as the number of considered neighbors increases, the value of NLL decreases toward the minimum value achieved when all data points are used. However, when the number of clusters is small, the minimum is achieved when a limited number of neighbors is considered.

One key difference between basin-hopping and the proposed optimization algorithm is visible from Fig. 3, where the dataset

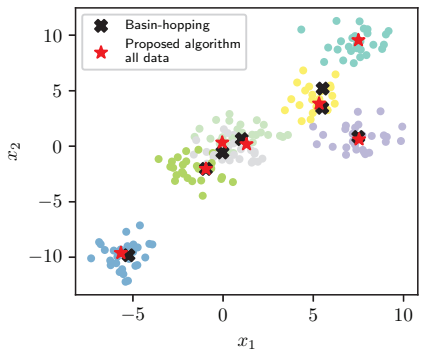


Fig. 3: Data and prototype distribution resulting from basin-hopping and the proposed optimization algorithm. Data points are colored according to their original subpopulation.

with seven clusters is depicted. Basin-hopping fails to detect the upper-right cluster, but it over-represented the yellow cluster with two prototypes. On the other hand, the proposed algorithm correctly identified all seven clusters, resulting in a minimal NLL.

B. Comparative evaluation on synthetic data

In this experiment, PFCM, FCM, and Gaussian Mixture Model (GMM) have been compared on synthetic data to show the main differences in their resulting clusterizations. The dataset consists of three clusters sampled from three Gaussian distributions with the variance equal to 1. For all the experiments, the methods were initialized with comparable values for the hyperparameter: as for FCM, the fuzziness parameter m was set to 2, which is equivalent to setting $\beta = 2$ for the PFCM; as for GMM, the variant with fully trainable covariance matrices was used.

To better assess the behavior of all the algorithms, we considered two scenarios, i.e., when the number c of clusters is less than or greater than three.

When the value c is less than the actual number of clusters in the data (Fig. 4), PFCM locates prototypes within one of the clusters, while both FCM and GMM locate one prototype in the middle of two clusters, actually not representing any data clouds.

The uncertainty in assigning a data point to a cluster can be measured in terms of entropy, evaluated as

$$H(\mathbf{x}) = -\sum_{i=1}^{c} \Pr\left(C = i|\mathbf{x}, \theta\right) \log_2 \Pr\left(C = i|\mathbf{x}, \theta\right) \qquad (9)$$

where $u_{ji} \equiv \Pr\left(C = i|\mathbf{x}_j, \theta\right)$ in the case of FCM. Data points can be marked with their entropy to show the distribution of uncertainty on data given the clustering results.

Fig. 4 shows a cloud of data with high uncertainty coming from the results of PFCM. By reporting the distribution of entropies, this cloud of uncertain data forms a subpopulation of entropies that can be easily detected and clearly shows the need to add more prototypes to better represent data. On the other hand, the distribution of entropies resulting from

TABLE I: Results for PFCM with basin-hopping and PFCM with the proposed algorithm. For both NLL and Execution Time, the results are the means over 10 runs, with the standard deviation in parentheses. Execution time is in seconds.

Dataset	2 Clusters		4 Clusters		7 Clusters	
	NLL	Time	NLL	Time	NLL	Time
Basin hopping	914.07 (1.11)	3.33 (0.86)	1003.01 (86.59)	15.34 (1.43)	795.25 (41.30)	28.28 (7.78)
k-NN with k=3	993.64 (40.26)	0.06 (0.04)	1034.31 (68.71)	0.30 (0.05)	844.14 (87.85)	0.56 (0.20)
k-NN with k=10	913.24 (0.00)	0.21 (0.04)	996.81 (83.21)	0.53 (0.11)	782.23 (19.89)	1.56 (0.75)
k-NN with k=30	913.24 (0.00)	0.34 (0.06)	995.74 (83.07)	1.20 (0.24)	768.38 (35.39)	2.37 (0.70)
k-NN with k=80	913.24 (0.00)	0.69 (0.02)	866.28 (0.00)	3.83 (0.80)	736.85 (23.35)	5.73 (0.71)
All data	913.24 (0.00)	1.76 (0.02)	866.28 (0.00)	6.61 (0.06)	720.11 (0.00)	12.09 (2.29)

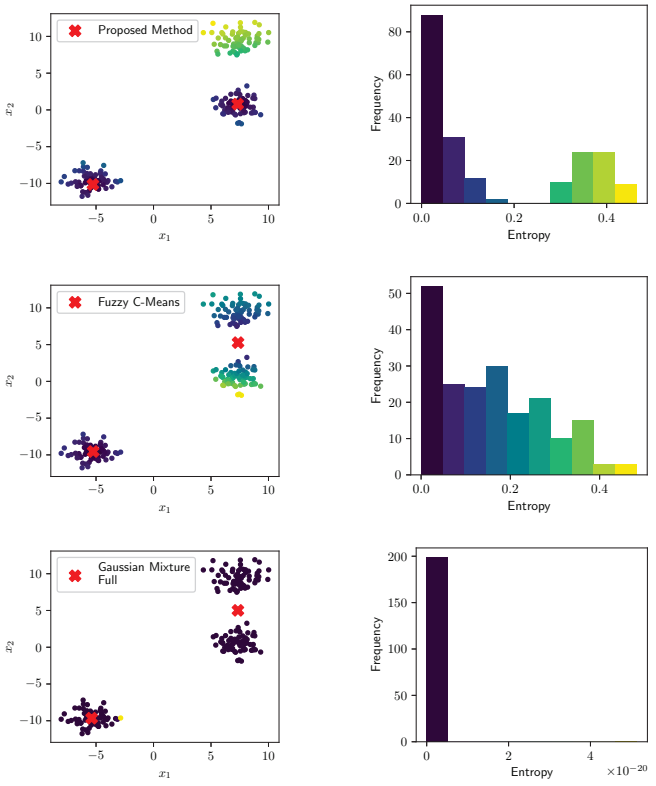


Fig. 4: Results of the three methods for $c = 2$. From top to bottom: PFCM, FCM, GMM. On the left: data distribution with the final prototype positions. (Data are colored according to their entropy.) On the right: distributions of entropies.

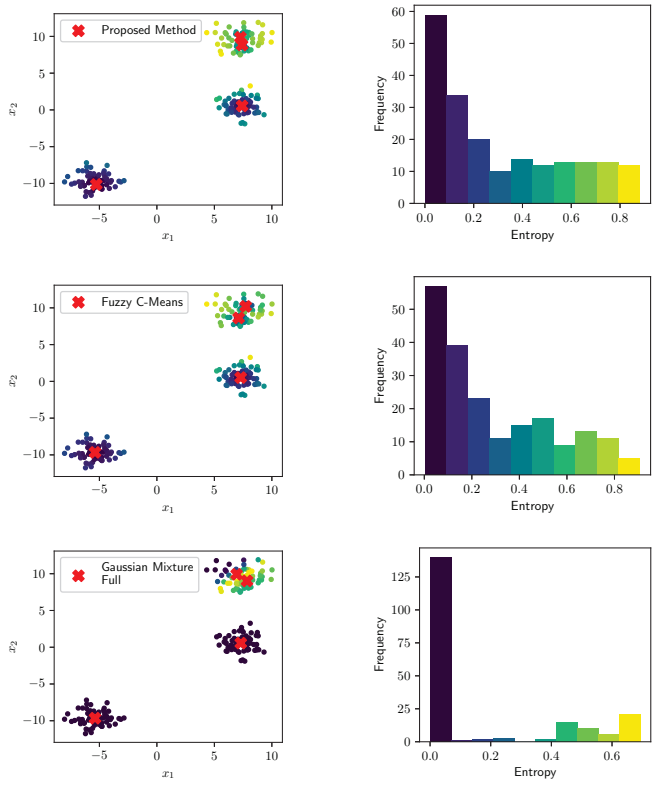


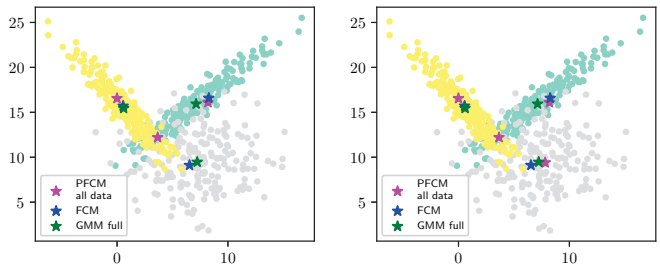
Fig. 5: Results of the three methods for $c = 4$. From top to bottom: PFCM, FCM, GMM. On the left: data distribution with the final prototype positions. (Data are colored according to their entropy.) On the right: distributions of entropies.

FCM makes the lack of an additional prototype less evident. In GMM, there is almost no uncertainty in assigning data to clusters, even though the lack of an additional prototype is apparent.

Fig. 5 shows the results of the methods when the number of prototypes is greater than the number of clusters in the data. All methods assign two prototypes to the top-right cluster; however, the uncertainty assigned to the data points is different: GMM assigns high uncertainty to the points between the two closest prototypes, while PFCM and FCM assign a more uniformly distributed uncertainty to all the points in the wrongly divided cluster.

In this case, a more uniformly distributed uncertainty on the points aligns with the possibility that the cluster might be incorrectly divided. In fact, as long as the distance of a data point from the prototypes increases, the uncertainty of assigning the point to one of the corresponding clusters increases. Since these data points belong to a cloud of data, such an increase of distance (and corresponding uncertainty) is evenly distributed. This behavior is different from that observed in Fig.4, where there is an observable "gap" in data corresponding to a separate cloud that is not represented by any prototype.

As a final test, we checked the results of the three clustering methods when data are generated from asymmetrical distributions (namely, Gaussian distributions with non-diagonal



(a) $c = 3$ (b) $c = 4$

Fig. 6: Resulting prototypes from PFCM (all data), FCM, and GMM on three overlapping clusters of data sampled from non-spherical distributions. The data points are colored based on their true label.

covariance matrices).

In Fig.6, the results are reported for GMM and FCM (with $c = 3$), and PFCM (for $c = 3$ and $c = 4$). In this case, the component density function (see Fig. 1) suggests a symmetrical assumption of data around a prototype, but this assumption is violated in this dataset. As a consequence, while GMM (which learns the full covariance matrix) can correctly locate the prototypes of the clusters, PFCM favors the representation of the densest region of the data space. When $c = 3$ PFCM leaves the sparsest cluster not represented, while it is recovered when $c = 4$. On the other hand, the averaging effect of FCM allows a localization of the prototypes that is close to GMM.

V. Conclusions

This paper presented an optimization algorithm for the PFCM method and compared it with the previously used basin-hopping algorithm, as well as with the well-known FCM and GMM clustering methods.

The experimental results on some synthetic data show a clear speed-up of PFCM with the proposed optimization algorithm over basin-hopping; this speed-up is motivated by two factors: (i) the proposed algorithm uses a subset of data points to update the prototypes, thanks to the k-NN procedure; (ii) the algorithm uses the available data points as possible locations for the prototypes without the need to search the entire data space.

Moreover, the final values of the objective function are generally lower than those obtained from basin-hopping (except when k is too small). This corroborates our hypothesis that the minimum value of the objective function is obtained when prototypes coincide with some representative data points (plus an offset that is required by the assumed probability distribution). Future investigation will be devoted to giving a theoretical grounding to this hypothesis.

The comparison of PFCM against FCM and GMM on controlled datasets shows that with PFCM it is potentially easier to detect the need to adapt the number of clusters to data. This can be accomplished by analyzing the distribution of class entropies calculated for each data point. However, in this work we only outlined this possibility, while we deserve a more in-depth analysis in a future work.

When data violate the assumption of symmetry around prototypes, the results of PFCM differ from those obtained from FCM and GMM. While this is not necessarily a drawback (PFCM tries to locate the prototypes on the densest parts of the data space), this result highlights the limitation of the current version of PFCM in assuming spherical clusters. Future developments will test PFCM using different, and possibly learnable, metrics that are used to define the probabilistic model.

Finally, PFCM was conceived after a reinterpretation of FCM in the context of probability theory. This gives clear semantics to the components of FCM, including the hyper-parameter m (equivalent to β in PFCM) that is connected with the variance of data within clusters or, equivalently, the expected distance from clusters [5]. Yet, this interpretation opens the door to further extensions that depart from the original version of FCM; in particular, as m is connected with the variance of data within clusters, it is possible to extend PFCM so that each cluster has its variance. However, this requires a structural change of the PFCM algorithm, which is the subject of future investigation.

Acknowledgment

The authors are GNCS-INdAM members. The research of Davide Cazzorla is funded by a Ph.D. fellowship within the framework of the project of the University of Bari Aldo Moro "Patti territoriali dell'alta formazione per le imprese" Ph.D. Project "Predictive AI in Value Based Procurement (VBP)", co-supported by InnovaPuglia S.p.A.

References

[1] T. J. Loftus, B. Shickel, J. A. Balch, P. J. Tighe, K. L. Abbott, B. Fazzone, E. M. Anderson, J. Rozowsky, T. Ozrazgat-Baslanti, Y. Ren, S. A. Berceli, W. R. Hogan, P. A. Efron, J. R. Moorman, P. Rashidi, G. R. Upchurch, and A. Bihorac, "Phenotype clustering in health care: A narrative review for clinicians," vol. 5, p. 842306, 2022.

[2] K.-S. Chuang, H.-L. Tzeng, S. Chen, J. Wu, and T.-J. Chen, "Fuzzy c-means clustering with spatial information for image segmentation," vol. 30, no. 1, pp. 9–15, 2006.

[3] S. Wang, "A study on the application of optimized k-means algorithm in intelligent hospital procurement supplier management," in *2023 7th International Symposium on Computer Science and Intelligent Control (ISCSIC)*. IEEE, 2023, pp. 266–270.

[4] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," vol. 10, no. 2, pp. 191–203, 1984.

[5] C. Mencar and C. Castiello, "A bayesian interpretation of fuzzy c-means," in *Fuzzy Logic and Technology, and Aggregation Operators*, S. Massanet, S. Montes, D. Ruiz-Aguilera, and M. González-Hidalgo, Eds. Springer Nature Switzerland, 2023, vol. 14069, pp. 443–454.

[6] D. Cazzorla and C. Mencar, "A soft clustering method derived from the probabilistic interpretation of fuzzy c-means," in *Advances in Fuzzy Logic and Technology*, M. Baczyński and *et al.*, Eds. Springer Nature Switzerland, 2025, vol. 15884, pp. 173–184, series Title: Lecture Notes in Computer Science.

[7] B. Olson, I. Hashmi, K. Molloy, and A. Shehu, "Basin hopping as a general and versatile optimization framework for the characterization of biological macromolecules," vol. 2012, pp. 1–19, 2012.

[8] M. B. Ferraro and P. Giordani, "Possibilistic and fuzzy clustering methods for robust analysis of non-precise data," *International Journal of Approximate Reasoning*, vol. 88, pp. 23–38, 2017.