A New K-Multiple-Means Clustering Method

Jingyuan Zhang(✉)

School of Computer Science, China University of Geosciences (Wuhan), Wuhan, China
jyz.cug@foxmail.com

Abstract. In the field of clustering, non-spherical data clustering is a relatively complex case. To satisfy the practical application, the solution should be able to capture non-convex patterns in data sets with high performance. At present, the multi-prototype method can meet the former requirement, but the time cost is still high. This paper proposes a new multi-prototype extension of the K-multiple-means type algorithm, which aims to further reduce the computation time in processing non-spherical data sets with a concise principle while maintaining close performance. Compared with other methods, the method still adopts the idea of multiple prototypes and uses agglomerative strategies in the phase of class cluster connection. However, to reduce the amount of data involved in the computation and the interference of incorrect partition, the subclass data of the first partition is filtered. In addition, the agglomeration is divided into two stages: the agglomeration between prototypes and the agglomeration between clusters, and two agglomeration modes are provided to deal with different clustering tasks. Before updating the means, the filtered data needs a quadratic partition. Experimental results show that compared with the state-of-the-art approaches, the proposed method is still effective with lower time complexity in both synthetic and real-world data sets.

Keywords: Unsupervised learning · K-multiple-means · Clustering · Agglomerative strategies · Quadratic partition

1 Introduction

Clustering is one of the basic problems in data mining and unsupervised learning. Under the assumption of data similarity, unlabeled data with high similarity are grouped into the same class with the metrics selected by users. So far, all kinds of clustering algorithms [1–7] emerge one after another, partition-based algorithm [8–15] is an important role. Among them, K-means [16] as a hard partition algorithm has become one of the most popular algorithms. In contrast, Fuzzy C-means [17], as a variant of K-means, is representative of the soft partition class algorithm. The latter has higher robustness because it takes into account the overlap between subclasses and transforms category membership into a degree of category membership.

G. Memmi et al. (Eds.): KSEM 2022, LNAI 13369, pp. 621–632, 2022.
https://doi.org/10.1007/978-3-031-10986-7_50

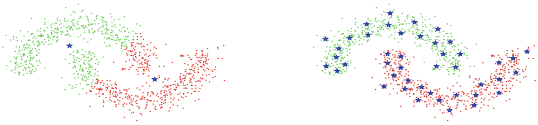
Although the K-means method performs well in spherical cluster data, lacks applicability for non-spherical data sets. There are two main research ideas: nonlinear clustering [26–30] and multi-prototype clustering [19–24,31,32]. The fundamental principle of nonlinear clustering is to map the original data by nonlinear technique and use the algorithms for clustering. For example, in kernel clustering and spectral clustering, the former uses a kernel function to map data into a certain feature space for a linear partition, and the latter uses low-dimensional embedding of similarity matrix generated by original data to obtain embedding vector for clustering. However, it is difficult to design suitable kernel functions or construction data graphs for each clustering problem. Another way is the multi-prototype method, which considers that each cluster can be represented by multiple prototypes, to better adapt to non-spherical data sets. Each prototype represents a subclass, which is still linear clustering. To achieve a specified number of clusters, it is necessary to merge subclasses, and the merging process is nonlinear clustering. At present, most methods adopt the aggregation strategy, but the selection of an appropriate combination point is not easy, which has a direct impact on the clustering effect. To merge multiple prototypes optimally on the global, a graph-based multi-prototype clustering algorithm was proposed. However, this method inevitably needs matrix calculation. When the data scale is large, matrix decomposition, spectral analysis and other operations can be very time expensive, so it needs further improvement. Aiming at the above defects, this paper presents a new K-multi-means extension method. The main contributions of this paper can be summarized as follows:

- To reduce the time cost of the algorithm and augment the feature of each subclass, part of the data is screened according to the distance relationship between the prototype and the data, so that the reserved data has better representativeness.
- The method in this paper divides the agglomerative strategy into two stages for decreasing the difficulty of the agglomeration: the merging between prototypes and the merging between clusters. In addition, to enhance the robustness of the algorithm, the proposed method provides two merging modes.
- The filtered data are likely to be partitioned incorrectly, so these data need a quadratic partition to improve the clustering effect of the algorithm.
- Experimental results show that the proposed method has similar performance to the state-of-the-art approaches with a lower time cost.

The remainder of this paper is organized as follows: Sect. 2 introduces the related work in this field. In Sect. 3, this paper describes the principle of the new k-multi-means extension and its computational complexity analysis. In Sect. 4, the experimental results and analysis are reported. Finally, the paper is summarized in Sect. 5.

2 Related Work

The multi-prototype method is an extension of traditional K-means to capture the non-convex pattern of data sets. As shown in Fig. 1, the data in the figure



(a) single-prototype (b) multi-prototypes

Fig. 1. The difference between single-prototype representation (a) and multi-prototypes representation (b). The blue pentagrams are prototypes. (Color figure online)

should be divided into two classes. The traditional K-means method only initializes the corresponding number of prototypes. As the features of the data cannot be captured, the data is incorrectly partitioned. The multi-prototype method initializes multiple prototypes and distributes them in the data. The data is divided into multiple independent subclasses, which can fully adapt to the shape and distribution of the data and discover non-convex patterns in the data.

Tao [19] generates multiple prototypes based on hierarchical subtractive clustering and then merges them according to the relationship between region density and subclass density of two subclasses. Liu et al. [20] proposed a simple multi-prototype clustering algorithm, which used squared-error clustering for splitting and then merged according to the density of overlapping regions of subclasses. Luo et al. [21] divided the data based on the minimum spanning tree and then merged the data according to the threshold value set by the user and the data distribution of the two subclasses. Ben et al. [22] proposed a multi-prototype method based on Fuzzy clustering. In the splitting stage, Fuzzy C-means and intra-cluster nonconsistency values were used to perform iterative dichotomy for subclasses, and then they were merged iteratively according to the overlap degree between clusters. Liang et al. [23] proposed a multi-prototype algorithm to deal with unbalanced distributed data. Reliable prototypes were obtained by Fast Global Fuzzy K-means, and the best-M Plot method was used to divide data. Finally, the grouping multi-center (GMC) algorithm is used to complete the merging. Nie et al. [24] proposed the multi-prototype algorithm based on the bipartite graph. Firstly, the distance relationship between the prototype and the data were used to construct the bipartite graph to obtain its Laplace matrix, and the clusters with a specified number were obtained by restricting the rank of the Laplace matrix [25]. This method transforms the clustering problem into an optimization problem, which is the state-of-the-art multi-prototype algorithm at present because of its superior clustering effect and relatively low time complexity.

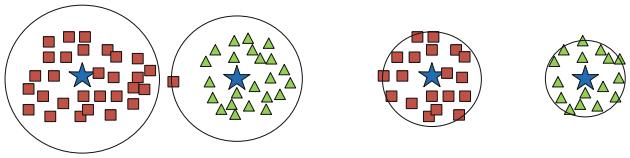
The clustering effect of the agglomerative method heavily depends on the subclass merging, if there is any error merging, all the subclasses of subsequent merging will be wrong, so the selection of connection points is a challenging problem. And the graph-based method can model the prototype and data globally, and delete part of the cut edges according to the feature matrix of the

graph, which belongs to the split strategy. In the next section, this paper will describes a clustering method using the aggregation strategy, which is close to the state-of-the-art method and has a lower time overhead.

3 The Proposed Clustering Algorithm

3.1 The New K-Multiple-Means

The method in this paper also includes two stages of splitting and merging. In the splitting stage, squared-error clustering is still used.



(a) Original (b) After screening

Fig. 2. Data filtering. The blue pentagram in the figure is the prototype, the red square and green triangle are two classes of data, and the circle represents the data division area. Figure (a) shows the original partition of the data, in which a red square is incorrectly partitioned to the prototype on the right. Figure (b) shows the division after screening, and the wrongly divided data has been eliminated. (Color figure online)

Due to the use of squared error clustering for division, the subclasses divided may have two types of data in one subclass. If the merging is performed directly, the accuracy will be greatly affected. Therefore, we have added a subclass data screening link. For each subclass, only those data with small absolute distance and large relative distance are retained, so that the retained data has a strong membership property to the prototype. See Fig. 2. The absolute distance is the distance between the data and the nearest prototype, and the relative distance is the difference between the distance between the data and the closest prototype and the distance between the next closest prototype. Denote a_i as the i-th data point, $C_{ij}$is the j-th nearest prototype to the data point a_i, $d(a_i, C_{ij})$ represents the distance between a_i and C_{ij}, then the membership degree $S(a_i)$ of the data point a_i is defined in the Eq. (1)

$$S(a_i) = \frac{d(a_i, C_{i2}) - d(a_i, C_{i1})}{d(a_i, C_{i1})} \tag{1}$$

After that, the median of all data points in the subclass is calculated, and only data points whose membership degree is not lower than the median are allowed to participate in the subsequent calculation.

In the merging stage, a certain number of prototypes that are as dispersed as possible are randomly selected as control points, and the remaining prototypes are merged. The control points can cover most of the prototypes of the class and naturally grow and expand according to the distance relationship between the data points and the prototypes, thus forming several clusters. The remaining prototypes require two rounds of voting to finally calculate which prototype should belong. The first round of voting is to obtain the most likely connected prototype of each data in the subclass, and calculate the connection probability p_{ij} of the data point a_i and its k nearest neighbors C_{ij} (see Eq. (2)), if there is a connection between the neighbors, that is, there are multiple neighbors that belong to the same cluster, the probability is accumulated.

$$p_{ij} = \frac{d(a_i, C_{ik+1}) - d(a_i, C_{ij})}{k * d(a_i, C_{ik+1}) - \sum_{j=1}^{k} d(a_i, C_{ij})} \tag{2}$$

Then, the connection probability of each data point is counted, and the probability of belonging to the same cluster is accumulated. The ratio of the sum of the connection probability of each cluster to the sum of the probability is the probability that the non-control prototypes match the corresponding prototype. See Eq. (3), $P(\alpha, \beta)$ is the probability that the unmatched prototype α merges with the prototype β, $p_{i,\beta}$ is the probability that the data point a_i in the unmatched prototype α is connected to the prototype β.The prototype with the highest connection probability is selected for merging to complete the second round of voting.

$$P(\alpha, \beta) = \frac{\sum_{i=1}^{l} p_{i\beta}}{\sum_{i=1}^{L} p_{ij}} \tag{3}$$

If the non-control prototype matches the control point, it will be merged directly. If it matches the non-control prototype, the two will be merged, and they will be used as the new control point, and then continue to merge. Therefore, in the process of merging non-control prototypes, the number of control prototypes will increase. This operation is to avoid prototypes that do not completely cover all clusters due to improper selection of control points. In addition, there may be some control points that are not merged with the prototypes, and do not play the role of control points. Therefore, when all non-control prototypes are completely merged, the isolated control points that have not been merged are merged with other control points, and the merging method is still using the second round of voting above.

When the non-control points are completely matched, several larger subclusters will be formed, usually, the number of clusters is greater than K. Therefore, further merging is needed to make the number of clusters reach K. At this time, the subclass cluster has formed a large scale, which contains rich cluster correlation information, and the method in this paper uses this information to merge. Referring to the relative inter-connectivity (Eq. (4)) and relative closeness (Eq. (5)) in the Chameleon algorithm [18] as the merging indicators, select the cluster with the largest value of Eq. (6) to merge until K clusters. In the

following equations, all connecting edges are included in the connecting edges between each data point and its k nearest neighbors.$\left|EC_{\{C_i,C_j\}}\right|$, $\bar{S}_{EC_{\{C_i,C_j\}}}$ is the sum and expectation of the probability of all cutting edges of the cluster C_i and the cluster C_j, $|EC_{C_i}|$, $\bar{S}_{EC_{C_i}}$ are the sum and expectation of all cutting edge probabilities inside the cluster C_i. $|C_i|$ is the number of data points in C_i.

$$RI\left(C_i, C_j\right) = \frac{2 * \left|EC_{\{C_i,C_j\}}\right|}{|EC_{C_i}| + \left|EC_{C_j}\right|} \tag{4}$$

$$RC\left(C_i, C_j\right) = \frac{\left(|C_i| + |C_j|\right) * \bar{S}_{EC_{\{C_i,C_j\}}}}{|C_i|\,\bar{S}_{EC_{C_i}} + |C_j|\,\bar{S}_{EC_{C_j}}} \tag{5}$$

$$f\left(C_i, C_j\right) = RI\left(C_i, C_j\right) * RC\left(C_i, C_j\right)^{\alpha} \tag{6}$$

Generally speaking, the above merging methods can achieve good results, especially when the distribution of the dataset is unbalanced, it is easy to find scarce classes with small sample sizes. However, when the data distribution is relatively balanced, but there are outliers in a certain category, the above-mentioned merging method will cause a wide range of wrong connections, and individual outliers may be classified into one category. To enhance the robustness of the algorithm, our method provides an additional merging mode. First, merge small-scale clusters according to the above indicators to form new clusters, until the current number of clusters $K' = K$, if $K' > K$, continue to merge clusters according to the indicators and don't care about cluster size anymore. Also, merges between new clusters are not allowed in merges. If $K' > K$, use the original merging method to merge larger clusters or new clusters. The user can choose the merging method according to the characteristics of the dataset. If there is a lack of understanding of the characteristics of the data set, a relatively stable merging method with a better clustering effect can be selected through multiple experiments. Hereinafter, the first merging mode is referred to as mode 1, and the latter is referred to as mode 2.

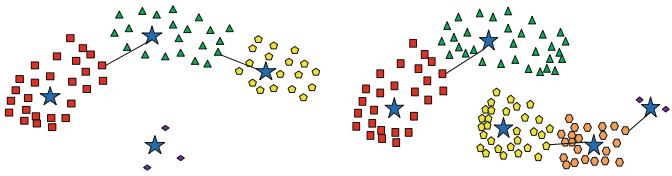


Fig. 3. The illustration of two merging modes. The graph meaning in the figure is the same as that in Fig. 2, where the connecting line represents the merging of two subclasses. In (a), merge only according to metrics, and small class clusters can be found. In (b), there is an extreme case in which there are only two samples of subclasses, so the subclasses with small scale are merged first to prevent the clusters combination of different classes from leading to a large-scale wrong partition.

After the cluster merging is completed, the data eliminated in the data screening stage needs to quadratic partition, and the wrongly divided data points are corrected. Using the above-mentioned first-round voting method, since there are only K clusters, the probabilities of the k nearest neighbors belonging to the same category are superimposed, and divided according to the accumulated probabilities, as shown in Eq. (7).

$$f\left(a_i\right) = \underset{C_j}{argmin}\left\{dis\left(a_i, \underset{\mathbf{C}}{argmax}\left\{\sum_{j=1}^{\widetilde{l}} \bar{p}_{ij}\right\}\right)\right\}, 0 < \widetilde{l} \leq k \tag{7}$$

In Eq. (7), $\bar{p}_{ij}$ is the connection probability between data a_i and a prototype C_j, $\mathbf{C}$ is a set of prototypes of a certain class, $\widetilde{l}$ is the number of prototypes of the a certain cluster.After the quadratic partition is over, the prototypes are updated according to $C_j = \frac{1}{l} * \sum_{i=1}^{l} a_{ij}$, where a_{ij} represents the data a_i divided into C_j, after which the next iteration is performed until the algorithm converges. See Algorithm 1 for a summary of the algorithm.

Algorithm 1 :The New K-Multiple-Means

Input: Data matrix $A \in R^{n*d}$, cluster number K, prototype number m, merging mode
Output: K clusters

```
Initialize multiple-means C_j.
Picking some prototypes as the control points at random.
while not converge do
    Partition all data by Squared-error clustering;
    For each j, screen data of C_j with Eq. (1);
    Merging the non-control points and outlying control points with Eqs. (2–3);
    while the number of clusters is greater than K do
        Merging the clusters with Eqs.(4–6) and selected merging mode;
    end while
    Quadratic partition for the filtered data with Eq. (7);
    For each j,update C_j;
end while
```

3.2 Computational Complexity

In this subsection, we will analyze the computational complexity of Algorithm 1. The time overhead of the initial partition of data is $O(nmd)$, n is the number of data, and d is the feature dimension of the data. The time complexity of the data filtering is $O(nm+\tilde{n}^2 m)$, and $\tilde{n}$ is the number of samples in the subclass. In addition, the time cost of selecting prototypes as the control points is $O(m\tilde{m}d)$, and $\tilde{m}$ is the number of control points. The time overhead of the merging stage includes non-control points merging and clusters merging, which is accumulated to $O(m\tilde{n}k^2 + nm^2 + nmk)$, where k is the number of neighbors. Quadratic partition requires $O(nmk^2)$ time cost. In summary, the total time complexity of the

method in this paper is $O(n(m^2 + mk + mk^2)t + \tilde{n}^2mt + nmdt)$, and t is the number of algorithm iterations.

When merging clusters, based on the k nearest neighbors distance relationships, only the nearest three clusters are selected for each cluster to calculate their weights respectively, and then they are merged according to the selected merging mode. Therefore, the method not only learns useful information but also eliminates the interference of errors, which further speeds up the running speed while ensuring the effect.

4 Experiments

In this section, we show the performance of the new K-Multiple-Means method on synthetic data and real benchmark datasets. For the convenience of description, the method in this paper is hereinafter referred to as nKMM.All the experiments are implemented in MATLAB R2020a, and run on a Windows 10 machine with 2.30 GHz i5-6300HQ CPU, 24 GB main memory.

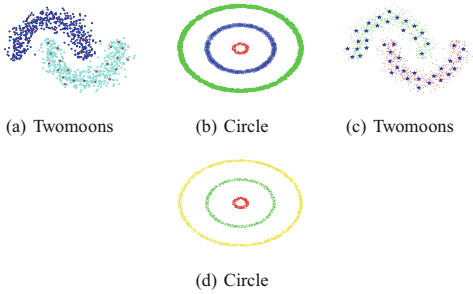
4.1 Experiments on Synthetic Data

Performance Comparison. This subsection mainly shows that the nKMM algorithm can process non-convex data sets, and its comparison method is KMM[1] There are two types of synthetic datasets used. The multi-prototype approach can be fully adapted to non-spherical data, capturing non-convex patterns. As shown in Fig. 4, the nKMM method can correctly partition the data and has the same performance as KMM. Both methods initialize the same number of prototypes, i.e., $m = \sqrt{n * K}$. The number of neighbors of nKMM is 5, and the number of neighbors of KMM is appropriately adjusted according to the data set. In addition, nKMM uses mode 1 for merging.

In the dataset Twomoons, the KMM algorithm incorrectly partitions the data at the intersection of the two classes of data, and these data are outliers for the subclass, and the KMM algorithm does not capture these subtle features. The nKMM algorithm improves this situation by partitioning the peripheral data twice.

Computation Time. Table 1 shows the specific information of the two datasets, as well as the performance of the KMM and nKMM algorithms on the two datasets. Since the number of iterations for each running of the algorithm is different, there will be a large error in calculating the total running time, so the average time of each iteration is calculated. In Table 1, the average time of each iteration of the algorithm is calculated, and nKMM has less time overhead than KMM.

[1] https://github.com/CHLWR/KDD2019_K-Multiple-Means.



(a) Twomoons (b) Circle (c) Twomoons

(d) Circle

Fig. 4. KMM:(a)(b), nKMM:(c)(d). In (a) and (c), the red points and blue pentagrams are prototypes. (Color figure online)

Table 1. Statistics of synthetic datasets and run time of clustering algorithms (s).

Datasets	Sample	Features	Clusters	KMM	nKMM
Twomoons	1000	2	2	0.146	0.11
Circle	3000	2	3	0.34	0.29

4.2 Experiments on Real Benchmark Datasets

Performance Comparison. To reflect the effectiveness of the method in this paper, the same real data set and evaluation index [24] are directly used, and other comparison algorithms are also the same.

Table 2. Statistics of real benchmark datasets and run time of multi-means clustering algorithms (s).

Datasets	Sample	Features	Clusters	KMM	nKMM
Wine	178	13	3	0.026	0.018
Ecoli	336	7	8	0.044	0.039
BinAlpha	1854	256	10	0.386	0.233
Palm	2000	256	100	0.819	0.430
Abalone	4177	8	28	1.237	0.718
HTRU2	17898	8	2	5.188	2.333

The experimental data includes Wine, Ecoli, Abalone, HTRU2, Palm, BinAlpha. The specific information of the data set is shown in Table 2. In addition to KMM, the comparison algorithms also include traditional K-means, SSC (Spectral Clustering), KKmeans (Kernel-based Clustering), RSFKC (Fuzzy Clustering), CLR (Graph-based Clustering), MEAP and K-MEAP (Exemplar-based Clustering). Evaluation indicators used Accuracy (ACC), Normalized Mutual Information

(NMI) and Purity. For different datasets, nKMM uses different merging methods and records stable experimental results. KMM and nKMM set the same number of prototypes and neighbors, $m = \sqrt{n * K}$, $k = 5$, the number of control prototypes in nKMM is $\tilde{m} = \sqrt{m * K}$. In addition, mode 1 is used in Ecoli and Abalone, and mode 2 is used for the other data sets. Since the K-means algorithm is more sensitive to initialization, to reflect the effect of the method, it is randomly initialized 100 times, and the average value of the evaluation index is recorded. The specific experimental data are shown in Table 3. The results of the CLR, MEAP and K-MEAP algorithms on HTRU2 are missing because the complexity of these three algorithms on large-scale datasets is too high. As can be seen from the table, nKMM achieves more than 90% of the performance of KMM with lower time complexity and is better than other algorithms.

Table 3. Clustering performance comparison on real-world datasets (%).

	Metric	K-means	SSC	KKmeans	RSFKC	CLR	MEAP	K-MEAP	KMM	nKMM	Percentage
Wine	ACC	94.94 (±0.51)	66.85	96.06 (±0.32)	95.50 (±3.72)	93.25	94.94	48.31	97.19 (±1.41)	93.95 (±2.75)	96.67
	NMI	83.23 (±1.53)	40.32	85.81 (±0.16)	84.88 (±4.57)	77.29	83.18	5.22	86.13 (±3.86)	81.21 (±5.03)	94.29
	Purity	94.94 (±0.51)	66.85	96.06 (±0.32)	95.50 (±1.71)	93.25	94.94	48.31	95.76 (±1.41)	93.94 (±2.75)	98.10
Ecoli	ACC	62.79 (±6.21)	59.82	34.52 (±1.16)	58.03 (±9.76)	52.38	42.55	74.10	78.85 (±4.46)	76.49 (±6.02)	97.01
	NMI	53.44 (±3.10)	54.80	25.92 (±1.85)	51.64 (±16.65)	53.08	44.12	58.77	69.48 (±4.86)	67.11 (±4.87)	96.59
	Purity	79.76 (±3.06)	82.33	61.30 (±3.00)	79.46 (±11.45)	79.76	42.55	80.41	82.37 (±3.95)	81.97 (±3.54)	99.51
BinAlpha	ACC	64.88 (±3.34)	66.82	28.26 (±0.74)	59.11 (±9.87)	67.40	40.99	62.94	68.87 (±7.00)	70.78 (±7.14)	**1**02.77
	NMI	62.81 (±1.87)	70.01	20.99 (±0.38)	61.95 (±13.25)	71.05	41.03	60.96	72.94 (±7.05)	72.58 (±3.35)	99.51
	Purity	72.33 (±2.82)	76.00	35.54 (±0.50)	71.19 (±11.96)	78.00	45.41	69.84	76.59 (±6.37)	76.18 (±4.83)	99.46
Palm	ACC	63.65 (±3.45)	59.78	68.70 (±0.83)	71.13 (±6.80)	68.65	71.55	40.20	76.40 (±2.21)	84.65 (±1.70)	**1**10.80
	NMI	87.55 (±1.08)	79.98	89.06 (±0.68)	89.82 (±8.51)	90.27	90.60	71.23	92.30 (±0.94)	95.63 (±0.39)	**1**03.61
	Purity	71.80 (±2.81)	62.90	74.60 (±0.46)	76.11 (±7.44)	79.45	77.80	45.70	81.75 (±1.66)	87.46 (±1.23)	**1**06.98
Abalone	ACC	14.62 (±0.88)	13.96	14.79 (±0.26) 26.43 (±0.34)	19.12 (±1.88)	14.96	19.70	16.51	20.20 (±1.02)	19.01 (±1.43)	94.11
	NMI	15.09 (±0.29)	14.37	14.76 (±0.14)	6.52 (±3.32)	15.07	7.53	15.52	16.03 (±1.75)	16.14 (±1.19)	**1**00.69
	Purity	27.36 (±0.63)	27.68	26.43 (±0.34)	19.89 (±2.08)	27.67	19.70	27.31	25.2 (±1.33)	25.02 (±1.13)	99.29
HTRU2	ACC	91.85 (±2.10)	92.22	59.29 (±1.20)	92.17 (±2.55)	–	–	–	95.49 (±2.21)	92.79 (±3.71)	97.17
	NMI	30.30 (±1.01)	34.90	7.97 (±0.56)	27.02 (±2.26)	–	–	–	40.12 (±1.55)	35.80 (±22.20)	89.23
	Purity	91.89 (±1.32)	93.35	90.84 (±0.78)	92.17 (±3.58)	–	–	–	95.49 (±1.92)	93.66 (±2.42)	98.08

Computation Time. The computation time comparison between nKMM and KMM is shown in Table 2. Synthetic datasets are inferior to real benchmark datasets in terms of feature dimension and number of samples, it fails to fully reflect the efficiency of the algorithm. In Table 2. The nKMM algorithm achieves a similar performance to the KMM algorithm with relatively lower time complexity and is even better than the KMM algorithm on Palm, which reflects that nKMM is effective and efficient.

5 Conclusion

This paper proposes a new multi-prototype extension with agglomerative strategies. The method filters the data after the first partition, only uses the data with obvious subclass features for calculation, and then partitions the filtered data twice. In addition, the method refines the previous agglomerative strategy, innovatively divides the merging into two stages: prototype merging and cluster merging, and provides two different merging modes for different datasets. The

experimental results show that the nKMM can achieve similar performance to the KMM with lower time complexity, and even achieve better results with half the time overhead on individual data sets, demonstrating the method is effective and efficient in this paper.

References

1. Dhillon, I.S.: Co-clustering documents and words using bipartite spectral graph partitioning. In: ACM SIGKDD (2001)
2. Jain, A.K., Narasimha Murty, M., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
3. Nie, F., Tian, L., Li, X.: Multiview clustering via adaptively weighted procrustes. In: ACM SIGKDD(2018)
4. Nie, F., Wang, X., Huang, H.: Clustering and projected clustering with adaptive neighbors. In: ACM SIGKDD (2014)
5. Von Luxburg, U.: A tutorial on spectral clustering. Statist. Comput. **17**(4), 395–416 (2007)
6. Von Luxburg, U.: Clustering stability: an overview. Found. Trends Mach. Learn. **2**(3) (2010)
7. Ben-Hur, A., Elisseeff, A., Guyon, I.: A stability based method for discovering structure in clustered data. Pac. Symp. Biocomput. 6–17 (2002)
8. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035. Society for Industrial and Applied Mathematics (2007)
9. Banerjee, A., Merugu, S., Dhillon, I.S., Ghosh, J.: Clustering with Bregman divergences. J. Mach. Learn. Res. **6**, 1705–1749 (2005)
10. Bezdek, J.C., Ehrlich, R., Full, W.: FCM: the fuzzy C-means clustering algorithm. Comput. Geosci. **2**(3), 191–203 (1984)
11. Cannon, R.L., Dave, J.V., Bezdek, J.C.: Efficient implementation of the fuzzy C-means clustering algorithms. IEEE Trans. Pattern Anal. Mach. Intell. **2**, 248–255 (1986)
12. Ding, C., He, X.: K-means clustering via principal component analysis. In: Twenty-First International Conference on Machine Learning, vol. 29. ACM (2004)
13. Hamerly, G., Elkan, C.: Learning the K in k-means. Adv. Neural Inf. Process. Syst. 281–288 (2004)
14. Pal, N.R., Bezdek, J.C.: On cluster validity for the fuzzy C-means model. IEEE Trans. Fuzzy Syst. **3**(3), 370–379 (1995)
15. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et. al.: Constrained k-means clustering with background knowledge. In: ICML, vol. 1, pp. 577–584 (2001)
16. MacQueen, J., et. al.: Some methods for classification and analysis of multivariate observations. In: Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, vol. 1, pp. 281–297 (1967)
17. Ruspini, E.H.: A new approach to clustering. Inf. Control **15**(1), 22–32 (1969)
18. Karypis, G., Han, E.-H., Kumar, V.: Chameleon: a hierarchical clustering using dynamic modeling. Computer **32**(8), 68–75 (1999). https://doi.org/10.1109/2.781637
19. Tao, C.-W.: Unsupervised fuzzy clustering with multi-center clusters. Fuzzy Sets Syst. **128**(3), 305–322 (2002)

20. Liu, M., Jiang, X., Kot, A.C.: A multi-prototype clustering algorithm. Pattern Recogn. **42**(5), 689–698 (2009)
21. Luo, T., Zhong, C., Li, H., Sun, X.: A multi-prototype clustering algorithm based on minimum spanning tree. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on, vol. 4, pp. 1602–1607. IEEE (2010)
22. Ben, S., Jin, Z., Yang, J.: Guided fuzzy clustering with multi-prototypes. In: IJCNN (2011)
23. Liang, J., Bai, L., Dang, C., Cao, F.: The K-means-type algorithms versus imbalanced data distributions. IEEE Trans. Fuzzy Syst. **20**(4), 728–745 (2012)
24. Nie, F., Wang, C.-L., Li, X.: K-multiple-means: a multiple-means clustering method with specified K clusters. In: ACM SIGKDD (2019)
25. Nie, F., Wang, X., Jordan, M.I., Huang, H.: The constrained Laplacian rank algorithm for graph-based clustering. In: AAAI (2016)
26. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: ACM SIGKDD (2004)
27. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: analysis and an algorithm. In: NIPS (2001)
28. Zha, H., He, X., Ding, C., Gu, M., Simon, H.D.: Spectral relaxation for k-means clustering. Adv. Neural Inf. Process. Syst. 1057–1064 (2002)
29. Bai, L., Liang, J.: A three-level optimization model for nonlinearly separable clustering. In: AAAI(2020)
30. Wang, C.D., Lai, J.H., Zhu, J.Y.: Graph-based multiprototype competitive learning and its applications. IEEE Trans. Syst. Man Cybern. C Appl. **42**(6), 934–946 (2012)
31. Wang, C.-D., Lai, J.-H., Suen, C.Y., Zhu, J.-Y.: Multi-exemplar affinity propagation. IEEE Trans. Pattern Anal. Mach. Intell. **35**(9), 2223–2237 (2013)
32. Wang, Y., Chen, L.: K-MEAP: multiple exemplars affinity propagation with specified K clusters. IEEE Trans. Neural Netw. Learn. Syst. **27**(12), 2670–2682 (2016)