

Received 1 November 2022, accepted 16 December 2022, date of publication 20 December 2022,  
date of current version 27 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3230935

## RESEARCH ARTICLE

# Initializing FWSA K-Means With Feature Level Constraints

ZHENFENG HE 

College of Computer and Data Science, Fuzhou University, Fuzhou 350108, China

e-mail: hezhenfeng@fzu.edu.cn

This work was supported in part by the National Natural Science Foundation of China under Grant 61976055, and in part by the Natural Science Foundation of Fujian Province under Grant 2022J01574.


**ABSTRACT** Weighted K-Means (WKM) algorithms are increasingly important with the increase of data dimension. WKM faces an initialization problem that is more complicated than K-Means' because in addition to picking initial cluster centers, it should also provide feature weights. Moreover, the one-dimensional solution to WKM's widely used objective function is unacceptable in most cases. Yet, the initialization of WKM, especially the initialization of feature weight, has been largely ignored. This paper studies the problem by analyzing Feature weight self-adjustment K-Means(FWSA K-Means), a popular WKM proposed to avoid the one-dimensional solution. Experimental results suggest that the algorithm is actually easy to cluster mainly based on a single feature information when it is not well initialized. Moreover, the paper argues that initial feature weights and cluster centers are equally important in determining the final partition. Therefore it suggests using feature level constraints to improve the initialization and proposes a semi-supervised algorithm Constrained FWSA K-Means (CFWSA K-Means). The algorithm uses constraints in evaluating feature weights and clusters to guide their evolution at the stage of initialization. Experimental results suggest that it is effective and robust in utilizing constraints. In addition, if its initialization process is started by the cluster centers provided by BRIK, an initialization approach for K-Means, the performance can be further improved.

**INDEX TERMS** Weighted K-Means, initialization, feature level constraint, semi-supervised clustering.

## I. INTRODUCTION

With the improvement of data acquisition, processing, and storage capacity, datasets contain more and more features. As different features often differ greatly in terms of their usefulness to a specified clustering task, they may not be treated equally. Clustering with feature selection and clustering with feature weighting have become hot research topics in the past 20 years [1], [2], [3], [4]. Allowing features closely related to the current clustering task to have a major influence, they often achieve an improvement in partitions.

Weighted K-Means (WKM), an important method of clustering with feature weighting, introduces feature weights, and evolves them in the clustering process to get high quality clusters, which are not necessarily spherical and fit the data

The associate editor coordinating the review of this manuscript and approving it for publication was Christos Anagnostopoulos .

better [5]. On the other hand, the addition of the feature weight makes both the decision model and the training process of WKM more complicated than K-Means'. As a result, it may be harder for ordinary users to get expected partitions.

The difficulty of WKM stems from K-Means. Being simple and useful, K-Means is recognized as one of the most popular clustering methods. Starting from some initial cluster centers, K-Means iteratively labels instances and updates cluster centers until the partition does not change. Yet, users are sometimes confused about the changeable partitions returned by the algorithm, because the final partition closely relates to the initial state, and some initial states may lead to unacceptable partitions [6]. This is the initialization problem of K-Means. WKM is an extension of K-Means, so it is expected that the choice of initial cluster centers will affect final partitions. However, the influence of the additional feature weight to the initialization is rarely addressed.

Deciding  $K$ , the cluster number, is sometimes considered as K-Means' initialization problem too. This paper considers only the problem whose  $K$  is fixed, so adaptively deciding  $K$  is not studied here.

### A. MOTIVATION

WKM is gaining more and more attention, many novel WKM algorithms have been being proposed and applied in various research or engineering areas over the past 10 years. As an extension of K-Means, WKM is expected to have an initialization problem. Yet, compared with a large number of studies on the initialization problem of K-Means, it is strange that there are few studies dedicated to WKM's initialization. One possible explanation for this phenomenon is: most people believe WKM and K-Means have similar initialization problems, and the methods for them are also similar. In other words, most people believe that initial cluster centers, not initial feature weights, have a decisive influence on WKM's final partition. However, this does not conform to the facts. Initial feature weights clearly affect final partition, so many researchers advise to initialize all features with an equal weight [3], [7].

Who has more influence on the final partition, initial feature weight or cluster center? How can we initialize wisely to decrease the possibility of getting the one-dimensional solution? We would study WKM's initialization by focusing on FWSA K-Means, and present an approach to improving the algorithm.

### B. CONTRIBUTIONS

The main contributions of this paper are as follows:

- (1) It suggests that feature weights and cluster centers are equally important in the initialization. The argument is proved through a simple example (section III B) and experiments on a synthetic dataset (section V B).
- (2) It suggests introducing feature level constraints into the initialization of FWSA K-Means, and presents a semi-supervised algorithm (section IV). The algorithm's performance is shown through the experiments on a synthetic dataset and 12 UCI datasets (section V).
- (3) It suggests a feature level constraint-based cluster quality evaluation approach to picking the worst cluster even in case of small clusters (section IV B).
- (4) It suggests FWSA K-Means, an algorithm proposed for avoiding the one-dimensional solution, may cluster mainly using one feature information when it is not carefully initialized. The argument is proved through experiments on a synthetic dataset (section V B).

## II. RELATED WORKS

There is a widespread concern over K-Means' initialization problem. Some users start K-Means randomly by setting initial cluster centers as arbitrarily selected instances, or the centers of randomly constructed clusters [8]. Such methods are simple, but often lead to bad results. Thus K-Means is often repeated several times to obtain multiple partitions,

of which the best one is picked [8], [9]. To further improve the initialization, various data distribution information is useful. Maximin method chooses the first cluster center randomly, then repeatedly selects the instance that is the farthest from the nearest selected centers to be the following cluster center [8], [10], [11]. On the other hand, the remaining  $K - 1$  centers may be selected in a probabilistic way instead. For example, K-Means++ selects the  $i$ -th instance with a probability  $d_i / (\sum_{i=1}^n d_i)$  as cluster center [12]. Here  $d_i$  is the distance between the  $i$ -th instance and its closest selected center; and  $n$  is the size of the dataset. Subspace information may also be utilized. For example, Onoda calculates  $K$  independent components (ICs) and then chooses the instances that are the closest to these ICs [13]. In addition, some methods pick initial cluster centers using cluster-level information. They often include a brief clustering process, which is independent from or combined into the following clustering process. PCA-Part starts by putting all instances in one cluster; then it iteratively splits the cluster with the greatest sum-squared-error in the direction of the cluster's largest eigenvector [14]. BRik clusters bootstrap replicas to get a large set of cluster centers, from which it picks the deepest ones [15]. Random swap and deterministic swap, which moderately change the partition by resetting a cluster center, enable continuous improvement of the initial state of K-Means [16]. Getting two partitions at first, COTCLUS improves initialization by simultaneously replacing two cluster centers, which are borrowed from another partition [17]. The latter two methods run an independent K-Means process after each round of randomly adjusting centers, so their initialization processes become the integral part of the evolution of partitions. Such continuous way is effective, which is also proved by Evolutionary K-Means approaches [18], [19].

Compared with K-Means, an additional weight vector is introduced by Weighted K-Means(WKM), yet how to initialize it is rarely addressed. For example, early works on WKM suggest starting with an equal feature weight without further analysis [3], [7]. After a few years, researchers find the one-dimensional solution. That is, when the exponent of feature weight is 1, there is an unacceptable solution: one feature's weight is 1 and the other features' weights are 0 [20]. As the equal-weight initial state is far from the solution, later works keep the same practice [21], [22], [23]. On the other hand, some WKM algorithms just start with arbitrary clusters without considering initialization [24]. As far as we know, there is no other research on the initialization of WKM's feature weight vector. To what extent can the selection of initial feature weights affect the final partition? This is still an open problem.

Background information can be helpful to the initialization of K-Means type algorithms. A few labelled instances can improve the quality of initial cluster centers (seeds) [25], [26], [27]. Instance-level constraints can be used to construct neighbourhoods of instances that are connected by constraints, then initial cluster centers are selected from these neighbourhoods [28].

There are numerous works on the initialization of K-Means type algorithms. Only the most related ones are introduced here. Interested readers can look for the introduction and comparison of more methods in [8], [9], and [29].

### III. THE INITIALIZATION PROBLEM OF FWSA K-MEANS

#### A. FWSA K-MEANS ALGORITHM

Suppose there is a set of  $p$ -dimensional instances  $D = \{x_1, x_2, \dots, x_N\}$ . Instance  $x_i$  is presented as  $(x_{i1}, x_{i2}, \dots, x_{ip})$ . Now we want to group them into  $K$  non-overlapping clusters  $C = \{C_1, C_2, \dots, C_K\}$ , whose centers are  $U = \{u_1, u_2, \dots, u_K\}$  respectively. An  $N \times K$  partition matrix  $L = (l_{ij})$  details the partition information:  $l_{ij} = 1$  when  $x_i \in C_j$ , otherwise  $l_{ij} = 0$ . As each instance  $x_i$  is assigned to one cluster, we have  $\sum_{j=1}^K l_{ij} = 1$ . In addition, there is a feature weight vector  $w = (w_1, w_2, \dots, w_p)$ . All weights are non-negative; and  $\sum_{i=1}^p w_i = 1$ .

In the clustering process, the objective function to be minimized is  $W$  in (1), the sum of the weighted distances between each instance and its corresponding cluster center. Here we only consider a simple case that all clusters share a subspace. Practically, cluster-wise feature weights are also often used, interested readers may refer to [5] and [30].

$$W = \sum_{r=1}^K \sum_{i=1}^N (l_{ir} \times dis(x_i, u_r)) \quad (1)$$

$$dis(x_i, u_r) = \sum_{j=1}^p (w_j \times (x_{ij} - u_{rj})^2) \quad (2)$$

$$u_{rj} = \frac{\sum_{i=1}^N (l_{ir} \times x_{ij})}{\sum_{i=1}^N l_{ir}} \quad (3)$$

Theoretically there is a solution of the feature weight to the minimization of (1): 1 for one element of  $w$ , and 0 for the other elements [20]. We may rewrite the objective function as (4).

$$W = \sum_{j=1}^p (w_j \times a_j) \quad (4)$$

$$a_j = \sum_{r=1}^K \sum_{i=1}^N (l_{ir} \times (x_{ij} - u_{rj})^2) \quad (5)$$

Because both  $w_j$  and  $a_j$  are non-negative, we have:

$$W \geq (\sum_{j=1}^p w_j) \times \min_j a_j = \min_j a_j \quad (6)$$

Therefore, given a partition,  $W$  is minimized if  $w_j = 1$  and  $w_j = 0, j \neq j'$ , where  $a_{j'} \leq a_j$ . This argument is the second part of Theorem 1 in [20], where interested readers may look for detailed analysis.

Such partition is clearly unacceptable in most cases, so there are various approaches to this problem. An exponent  $\alpha$  ( $\alpha < 0$  or  $\alpha > 1$ ) could be added to feature weight [20]. However, choosing a suitable and explainable  $\alpha$  is not

easy. Another method is adding an item to (1). For example, the entropy of feature weights or cluster sizes can be added [31], [32]. These methods actually complicate the model. In addition, how to balance the influence of original objective function and the added item is an open problem. Comparatively, finding a local solution without changing the model is simpler.

FWSA K-Means is a popular method that evolves the solution of  $W$  in an iterative way [21]. The algorithm sets initial feature weight vector as  $(1/p, 1/p, \dots, 1/p)$ . At each iteration, it adjusts feature weights using (7), where  $\Delta w_j$  is the adjustment margin of the  $j$ th feature.

$$w_j^{(new)} = \frac{1}{2}(w_j^{(old)} + \Delta w_j) \quad (7)$$

$$\Delta w_j = \frac{b_j/a_j}{\sum_{i=1}^p (b_i/a_i)} \quad (8)$$

$$b_j = \sum_{r=1}^K ((\sum_{i=1}^N l_{ir}) \times (u_{rj} - g_j)^2) \quad (9)$$

$$g_j = \frac{\sum_{i=1}^N x_{ij}}{N} \quad (10)$$

where  $a_j$ , computed by (5), is the sum of the squared differences between each instance and its corresponding cluster center according to the  $j$ th feature; and  $b_j$ , computed by (9), is the weighted sum (here the weight is cluster size) of the squared differences between each cluster center and the global center according to the  $j$ th feature. A comparatively larger  $b_j/a_j$  suggests clusters are well separated according to the  $j$ th feature, so its weight should be larger.

FWSA K-Means is based upon the iterative weight updating process (Algorithm 1).

---

#### Algorithm 1 FWSA K-Means Algorithm [21]

---

**Input:**

- a set of instances  $D$ ;
- cluster number  $K$ ;

**Output:**

- a set of cluster centers  $U$ ;
  - feature weight vector  $w$ ;
- 1: initialize cluster centers  $U$
  - 2: initialize feature weight vector  $w = (1/p, 1/p, \dots, 1/p)$
  - 3: **while** termination condition unsatisfied **do**
  - 4:   based upon  $U$  and  $w$ , compute partition matrix  $L$
  - 5:   based upon  $L$ , update cluster center  $U$
  - 6:   based upon  $U$  and  $L$ , compute weight change  $\Delta w$ , then update  $w$
  - 7: **end while**
- 

The first 2 steps of Algorithm 1 initialize cluster centers and feature weights. After that, the algorithm iteratively updates the partition and the feature weight vector. Step 4 and 5 are similar to that of K-Means. At step 4, distances between every instance and every cluster center are computed using (2); then each instance is assigned to its

closest cluster. At step 5, the center of each cluster is updated to the mean of the instances from the cluster using (3). At step 6, a feature weight change is calculated using (8), and then the feature weight is updated using (7). As the value of the objective function in (1) strictly decreases in the process, FWSA K-Means converges after several iterations [21]. A commonly used termination condition at step 3 is that  $L$  does not change and  $w$ 's change is marginal in the last iteration.

**B. INFLUENCE OF FWSA K-MEANS' INITIAL FEATURE WEIGHTS**

FWSA K-Means has an initialization problem. In fact, this argument is intuitively easy to get: when we set the initial state with final partition and the corresponding one-dimensional feature weight solution, the algorithm stops at the point because the algorithm strictly decreases  $W$  in each iteration.

We will further analyse FWSA K-Means' initialization problem experimentally with a toy example in figure 1. The dataset has two different 2-cluster partitions, which are shown in the two sub-figures.

Partition A is shown in the left sub-figure. One cluster is at the top-left, and the other one is at the bottom-right. The top-left cluster consists of 135 red points, distributed in three lines. The leftmost line has 39 points:  $(-2, -0.95), (-2, -0.9), (-2, -0.85), \dots, (-2, 0.95)$ . The rightmost line has 39 points too, but their  $x$  values are changed to 2. The middle line has 57 points:  $(0, -1.4), (0, -1.35), (0, -1.3), \dots, (0, 1.4)$ . Then we move the three lines right by 2 and down by 3. As a result, we get 135 black points distributed in 3 lines. Two cluster centers are shown with green solid dots.

The right sub-figure presents another possible partition—partition B: one cluster on the left, and the other on the right. Cluster centers are also shown in green solid dots.

As there are totally 2 features and the sum of their weights is fixed at 1, we need only give the weight of the first feature. Here and hereafter in this subsection: when we consider feature weight, only the weight of the first feature is presented.

Firstly we test the influence of initial feature weight to final partitions based on the evolution of first feature weight. The results are shown in figure 2. Left sub-figure presents the evolution starting from partition A, that is, the initial cluster centers are:  $(0, 0)$  and  $(2, -3)$ . And the right sub-figure shows the evolution starting from partition B.

We test 11 initial feature weights, which are  $0, 0.1, 0.2, \dots, 1$ . After several iterations, FWSA K-Means converges. Yet, the feature weight converges at two different points, which are shown as dashed red lines. The bottom line, which is about 0.085 at the first feature weight, corresponds to partition A, whereas the top line, about 0.945 instead, corresponds to partition B. So the left sub-figure suggests that an initial feature weight not less than 0.7 changes the partition from A to B. And the right sub-figure tells a similar story: when the initial first feature weight is not bigger than 0.4, the partition changes from B to A. In other words, when initial

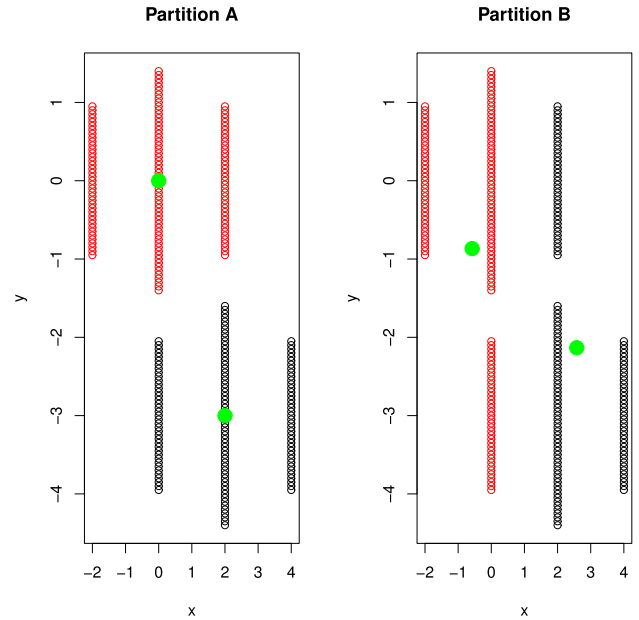


FIGURE 1. Two possible 2-cluster partitions of a dataset.

cluster centers are fixed, we may change initial feature weight to get an expected partition.

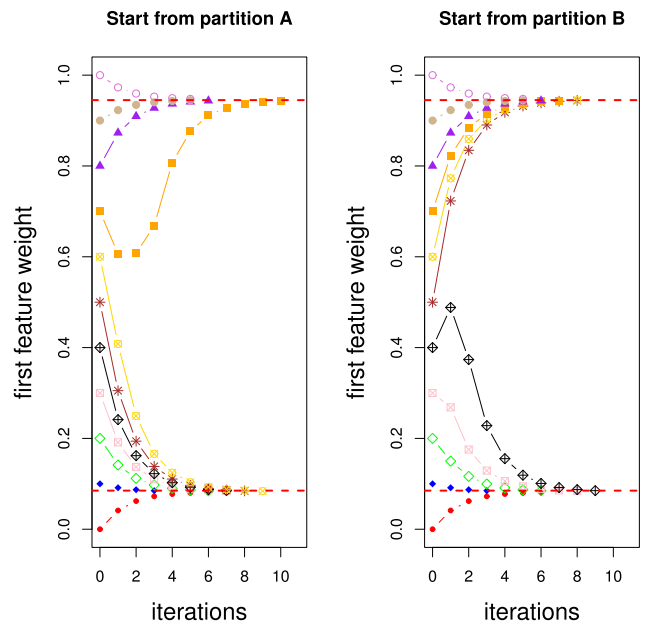


FIGURE 2. Evolution of the weight of the first feature.

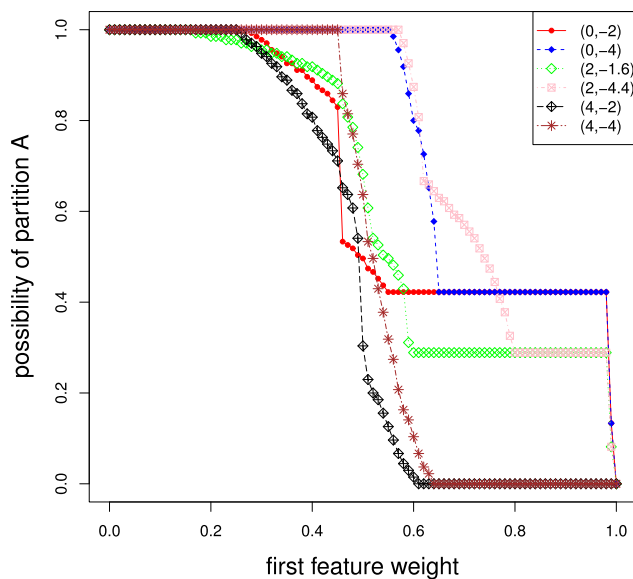
Secondly we evaluate the joint influence of the initial cluster center and the initial feature weight to final partition from the possibility of converging to partition A in the case of a given initial feature weight value.

We test totally 101 different initial feature weights, which are  $0, 0.01, 0.02, \dots, 1$ . For each weight, a black point and a red point from partition A are selected as initial cluster centers. We constantly use a black point as one initial cluster

center, but choose the other initial center from the 135 red points. In this way, for each weight, we test 135 different sets of initial cluster centers. As a result, each point of the curve in figure 3 is the possibility of converging at partition A in the 135 experiments.

The constantly selected initial center is located at the end area of a black line segment of partition A. In other words, we have tried 6 different constant initial cluster centers, which are  $(0, -2)$ ,  $(0, -4)$ ,  $(2, -1.6)$ ,  $(2, -4.4)$ ,  $(4, -2)$ ,  $(4, -4)$ . Therefore, there are six curves in figure 3. For example, when initial feature weight is 0.4. We always get partition A when the constantly selected initial cluster center is  $(0, -4)$ ,  $(2, -4.4)$  or  $(4, -4)$ . The possibility decreases to about 0.9 when the center is  $(2, -1.6)$  or  $(0, -2)$ . And the possibility further decreases to about 0.8 when the center is  $(4, -2)$ .

The situation of an empty cluster hardly ever happens in the experiments, so its influence to final possibility is marginal. When there is a null cluster, we randomly select a point as the cluster's center and continue iterations.



**FIGURE 3.** Possibility of converging at partition A under different initial feature weights and initial cluster centers.

From figure 3, it is clear that both initial cluster centers and feature weights significantly influence the final partition. Firstly, when the initial first feature weight is between 0.2 and 1, different initial cluster centers may lead to different final partitions. Secondly, the increase in initial first feature weight clearly decreases the possibility of having partition A as the final partition.

We all know that initial cluster centers are important for K-Means type algorithms, but the experiments in this subsection show that initial feature weight is also important:

(1) Initial feature weight significantly influences final partition. In both figure 2 and figure 3, FWSA K-Means may change its output when feature weight changes. Even when

initial cluster centers are true cluster centers, some initial feature weights change the final partition (figure 2).

(2) Initial feature weight closely relates to the final weight. In other words, a small initial weight often leads to a not big final weight, whereas a big weight usually causes a not small final weight. This is obvious in figure 2. On the other hand, figure 3 suggests a small initial feature weight often leads to partition A. As partition A corresponds to the case that the first feature weight is 0.085, which is far less than partition B's 0.945, figure 3 actually gives the same judgement.

## IV. FWSA K-MEANS WITH FEATURE LEVEL CONSTRAINTS

### A. CLUSTERING WITH FEATURE LEVEL CONSTRAINTS

Though clustering is generally considered to be unsupervised, it often needs some background knowledge to get an acceptable partition. Given a dataset, usually there are multiple plausible partitions, each of which may be expected by some users. How to find a segmentation expected by current users is a challenging problem. For subspace clustering, the problem becomes more difficult and deserves even more attention because the variability of subspace greatly increases the number of plausible partitions. For example, the earth, the moon, Jupiter, and Saturn are celestial bodies in solar system. How should we cluster them, astronomers and astronauts may give different answers:

(1) Astronomers tend to cluster the earth, Saturn and Jupiter into a group because they all move around the sun, whereas the moon moves around the earth.

(2) Astronauts would group the moon, Jupiter and Saturn together, as they are possible destinations of space flight missions, whereas the earth is the place of departure.

For such a simple clustering task, different people have different expectations because they pay attention to different features. Astronomers pay more attention to the motion mode of celestial bodies, while astronauts only care about whether the celestial bodies are a suitable destination for an interstellar travel. Clearly, it is important for subspace clustering algorithms to know which features are more interesting to current users.

Moreover, the experiments in the last subsection suggest initial feature weight closely relates to the final weight. If we want a feature to have a big weight, we may initialize it with a big weight.

Feature level constraint, also named as feature order constraint or attribute level constraint, provides constraints on the feature weight vector  $w$  by directly describing the relative importance on some feature-pairs. The constraint set is a group of 3-tuples  $Z = \{z_1, z_2, \dots, z_c\}$ , each of which defines the minimum difference between two elements of  $w$  [33], [34]. For example,  $z_i = (i_1, i_2, \delta_i)$  suggests that users expect  $w_{i_1} - w_{i_2} \geq \delta_i$ . Here  $i_1, i_2 \in \{1, 2, 3, \dots, p\}$ ,  $i_1 \neq i_2$ , and  $\delta_i \in [0, 1)$ .

Given a set of feature level constraints  $Z$ , we use (11) to evaluate a feature weight vector  $w$ . The algorithm thinks  $w^a$

is better than  $w^b$  if and only if  $f(w^a) > f(w^b)$ .

$$f(w) = \sum_{z_i \in Z} \min(w_{i1} - w_{i2} - \delta_i, 0) \quad (11)$$

Recently another feature level constraint is proposed. It uses a preference vector, which is the expected feature weight. Besides, a confidence level is introduced to enable users to express their confidence to the vector [35]. As it is hard for users to give the whole vector, in this paper, we would use the 3-tuple based constraints.

### B. PARTIAL WEIGHT CHANGE BASED CLUSTER SELECTION

K-Means type algorithms can be improved by adjusting the center of the worst cluster. For example, Evolutionary K-Means algorithms usually evolve partitions by splitting or removing the worst cluster [18], [19]. To improve the initialization of FWSA K-Means, we would adjust the center of the worst cluster.

To pick the worst cluster, we use cluster-wise feature weight to evaluate each cluster's contribution separately. Both  $a_j$  and  $b_j$  can be regarded as the cumulative result of clusters. We can see this more clearly by changing (5) to (12), and replacing (9) with (14).

$$a_j = \sum_{r=1}^K a_{rj} \quad (12)$$

$$a_{rj} = \sum_{i=1}^N (l_{ir} \times (x_{ij} - u_{rj})^2) \quad (13)$$

$$b_j = \sum_{r=1}^K b_{rj} \quad (14)$$

$$b_{rj} = (\sum_{i=1}^N l_{ir}) \times (u_{rj} - g_j)^2 \quad (15)$$

Based upon  $a_{rj}$  and  $b_{rj}$ , we may use (16) to compute cluster-wise feature weight, or cluster-wise feature weight change. The two cluster-wise indices were actually proposed for a WKM approach, which computed a cluster-wise feature weight change using (17) [30].

$$\Delta \hat{w}_{rj} = \frac{b_{rj}/a_{rj}}{\sum_{i=1}^p (b_{ri}/a_{ri})} \quad (16)$$

$$\Delta w_{rj} = \frac{b_{rj}/(a_{rj} + \delta)}{\sum_{i=1}^p [b_{ri}/(a_{ri} + \delta)]} \quad (17)$$

Here  $\delta$  is a small positive constant added to prevent the denominator from being extremely small, which is very likely to happen in case of small clusters. However, it is hard to choose a suitable  $\delta$ . When it is small, small clusters may have an unacceptable influence on feature weight. Yet, a big  $\delta$  may make the feature weight insensitive to the change in small clusters.

In order to avoid using  $\delta$  and make the approach robust to small clusters, in stead of picking the worst cluster directly,

we choose the cluster whose removal leads to the best feature weight. Given a  $K$ -cluster partition, after removing cluster  $r$ , the weight change computed on the remaining instances is shown as (18). As  $\Delta \hat{w}_j^{(r)}$  calculates weight change on  $K - 1$  clusters, we call it *partial weight change*. When  $\Delta \hat{w}_j^{(r)}$  is the best according to given feature level constraints (using (11)), the  $r$ th cluster is considered as the worst cluster, which should be changed significantly. Thus we randomly select an instance as the cluster's new center.

$$\Delta \hat{w}_j^{(r)} = \frac{\hat{b}_j^{(r)}/\hat{a}_j^{(r)}}{\sum_{i=1}^p (\hat{b}_i^{(r)}/\hat{a}_i^{(r)})} \quad (18)$$

$$\hat{a}_j^{(r)} = a_j - a_{rj} \quad (19)$$

$$\hat{b}_j^{(r)} = b_j - b_{rj} \quad (20)$$

### C. CONSTRAINED FWSA K-MEANS

Initializing feature weight and cluster centers based upon give constraints, Constrained FWSA K-Means algorithm is as follows.

CFWSA K-Means incorporates constraints in the initialization process (step 1 to 19).

From step 1 to 5 of Algorithm 2, feature weight vector is initialized. At step 3, new feature weight is generated by adding Gaussian noise in (21) and normalizing in (22). It replaces the current feature weight when it is better (step 5). In our experiments, we set  $\sigma = \alpha/p$ , where  $p$  is the number of features and  $\alpha = 0.1$ . In the algorithm, function  $f$  evaluates feature weight using (11).

$$\bar{w}_j = \max(0, w_j + N(0, \sigma)) \quad (21)$$

$$\tilde{w}_j = \frac{\bar{w}_j}{\sum_{i=1}^p \bar{w}_i} \quad (22)$$

From step 6 to 20, the algorithm iteratively evolves initial cluster centres. Firstly partition and feature weight are adjusted using one step FWSA K-Means (step 8 to 10). Then partial weight changes  $\Delta \hat{w}^{(1)}, \Delta \hat{w}^{(2)}, \dots, \Delta \hat{w}^{(K)}$  are computed (Step 12). At step 13, the cluster that leads to the best partial weight change is selected. If the change is better than current weight change, it becomes current weight change (step 15), and the center of the cluster will be replaced by a randomly selected instance (step 16).

To make the partial weight change estimation robust to noise, too big cluster is ignored. Sometimes there is a big cluster, which is so big that removing it means ignoring most instances and the remaining instances can not make a robust evaluation of the partial weight change. Such cluster is ignored (step 11 to 12). Here  $C_i$  is the  $i$ -th cluster; and  $|C_i|$  is the size of the cluster.

At step 6, the initialization of cluster centers is started randomly. Yet, we may use initialization methods for K-Means to get better cluster centers.

### V. EXPERIMENTS

The algorithms are compared on a synthetic dataset and 12 UCI datasets.

**Algorithm 2** Constrained FWSA K-Means(CFWSA K-Means) Algorithm

**Input:**

- a set of data instances  $D$ ;
- cluster number  $K$ ;
- a set of constraints  $Z$ ;
- rounds of adjusting initial feature weight  $N_w$ ;
- rounds of adjusting initial cluster centers  $N_c$ ;

**Output:**

- cluster centers  $U$ ;
- feature weight vector  $w$ ;
- 1: set initial feature weight vector  $w = (1/p, 1/p, \dots, 1/p)$
- 2: **for**  $i=1$  to  $N_w$  **do**
- 3:     randomly generate  $\tilde{w}$  by adding Gaussian noise to  $w$
- 4:     **if**  $f(\tilde{w}) > f(w)$  then set  $w = \tilde{w}$
- 5: **end for**
- 6: randomly set initial cluster centers  $U$
- 7: **for**  $iter=1$  to  $N_c$  **do**
- 8:     based upon  $U$  and  $w$ , compute partition matrix  $L$
- 9:     based upon  $L$ , update cluster center  $U$
- 10:    based upon  $U$  and  $L$ , compute weight change  $\Delta w$
- 11:    **for** ( $i = 1$  to  $K$ ) **if** ( $|C_i| < |D|/2$ ) then add  $i$  to  $S$
- 12:    **for** ( $r$  in  $S$ ) compute  $\Delta \hat{w}^{(r)}$  using (18)
- 13:    set  $r = \arg \max_{r \in S} f(\Delta \hat{w}^{(r)})$
- 14:    **if**  $f(\Delta \hat{w}^{(r)}) > f(\Delta w)$  **then**
- 15:     set  $\Delta w = \Delta \hat{w}^{(r)}$
- 16:     sample an instance as the center for cluster  $r$
- 17:    **end if**
- 18:    update  $w$  using (7)
- 19: **end for**
- 20: **while** termination condition unsatisfied **do**
- 21:     based upon  $U$  and  $w$ , compute partition matrix  $L$
- 22:     based upon  $L$ , update cluster center  $U$
- 23:     based upon  $U$  and  $L$ , compute weight change  $\Delta w$ , then update  $w$
- 24: **end while**

**A. GENERATION OF FEATURE LEVEL CONSTRAINTS**

$\Delta w_j$ , computed with (8), is the expected weight of the  $j$ -th feature, so we use it as the “true” feature weight for constraint generation.

We generate constraints using the method adopted by [33]. Firstly features are sorted according to their weights. To generate a feature level constraint  $z = (x, y, \delta)$ , we randomly select a feature  $x$  from the top 50% features, and a feature  $y$  from the bottom 50% features respectively. Then set  $\delta = w_x - w_y$ , here  $w_x$  and  $w_y$  are the weights of feature  $x$  and  $y$ .

Each constraint is generated independently. That is, new constraints are generated without considering the constraints generated, so they are generally not independent of each other, and may even be repeated. All constraints are applied to evaluate constraints using (11).

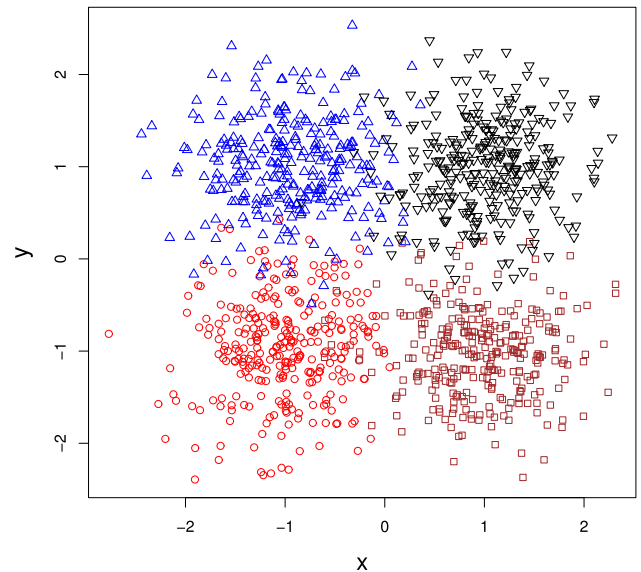
In the second rounds of experiments on UCI datasets, noisy constraints are applied. Firstly, we get the true order

of features according to their weights. Then  $p$  uniformly distributed positive random numbers are generated, sorted and assigned to corresponding feature according to the order ( $p$  is the data dimension). At last, the feature weights are normalized so their sum is 1. In this way, features keep the true order. That is, if feature A’s true weight is bigger than feature B’s. Then A’s noisy weight will not be less than B’s noisy weight.

**B. EVALUATION ON SYNTHETIC DATASETS**

1) DATASET

We test algorithms on a synthetic dataset. It has 4 clusters, each containing 300 instances. The dataset has 4 features, but only the first 2 features are informative. Expected cluster centers are  $(-1, -1, 0, 0)$ ,  $(-1, 1, 0, 0)$ ,  $(1, -1, 0, 0)$  and  $(1, 1, 0, 0)$ . Zero-mean Gaussian noise with a standard deviation  $\sigma = 0.5$ , is added to the centres to generate instances. The data distribution of the first 2 dimensions is show in figure 4.



**FIGURE 4.** First two dimensions of the synthetic dataset.

2) EXPERIMENT ON THE QUALITY OF RETURNED FEATURE WEIGHT

Firstly, we would compare the algorithms’ performance of finding true feature weight. Using different seeds to initialize random number generator, we get 1000 replicas of the dataset. Then two initial feature weight vectors and two sets of initial cluster centers are tested.

The feature weights are:

$$w_a = (0.5, 0.5, 0, 0) \text{ and } w_b = (0.25, 0.25, 0.25, 0.25).$$

Clearly  $w_a$  can be recognized as the true feature weight.

The initial cluster center sets are:

$$u_a: \{(-1, -1, 0, 0), (-1, 1, 0, 0), (1, -1, 0, 0), (1, 1, 0, 0)\}$$

$$u_b: \{(0, 1, 0, 0), (0, -1, 0, 0), (1, 0, 0, 0), (-1, 0, 0, 0)\}$$

Clearly  $u_a$  can be recognized as the true cluster centers.

For FWSA K-Means, the initial state is a combination of a feature weight vector and a set of cluster centers, so there are  $2 * 2 = 4$  possible initial states. In addition, the initial state suggested by [21] is also tested.

We also test CFWSA K-Means on the replicas. Two constraints are incorporated:  $(1, 3, \delta)$ ,  $(2, 4, \delta)$ , here two different  $\delta$ s have been tried.

Thus 7 different methods are compared:

State A (FWSA K-Means): using  $w_a$  as initial feature weight vector, and  $u_a$  as initial cluster centres.

State B (FWSA K-Means): using  $w_a$  as initial feature weight vector, but  $u_b$  as initial cluster centres.

State C (FWSA K-Means): using  $w_b$  as initial feature weight vector, and  $u_a$  as initial cluster centres.

State D (FWSA K-Means): using  $w_b$  as initial feature weight vector, and  $u_b$  as initial cluster centres.

State E (FWSA K-Means): using  $w_b$  as initial feature weight vector, and 4 randomly selected instances as initial centres.

State F (CFWSA K-Means):  $\delta = 0.4$ , that is, constraints  $(1, 3, 0.4)$  and  $(2, 4, 0.4)$  are applied.

State G (CFWSA K-Means):  $\delta = 0.2$ , that is, constraints  $(1, 3, 0.2)$  and  $(2, 4, 0.2)$  are applied.

The cumulative distributions of the maximum feature weight are shown in figure 5. When the maximum feature weight is close to 1, the data is clustered mostly based on one feature. Yet, there are 2 informative features, so the curve should not rise significantly at the right end.

The cumulative distributions of the weight of the first feature are presented in figure 6. The true weight should be close to 0.5. Thus the expected curve is staying at zero when feature weight is much smaller than 0.5; increasing sharply near 0.5; and staying at 1 when it is much bigger than 0.5. In other words, the curve should have a sharp increase in the middle, and be smooth otherwise.

The figures suggest, in terms of final feature weights, different initial states lead to a huge gap in the quality of partitions:

(1) Initial state has a significant impact on the final partition, and good initial cluster center *seems to be* more important than good feature weight. State A assigns both the true feature weight and the true cluster centres, so it completely avoids the single dimensional solution (Figure 5). State C provides the true cluster centres only, yet it avoids the solution too. In addition, both states make the first feature weight close to 0.5 (Figure 6).

(2) Adding feature level constraints into the initialization process improves the clustering result. The feature weight returned by CFWSA K-Means (State F and G) is better than the cases when bad or random cluster centres are provided (State B, D and E).

(3) Strict constraints make a difference. Constraint with a bigger  $\delta$  is a stricter constraint because it is harder to be satisfied. In figure 6, when  $\delta = 0.4$  (State F), the returned first feature weight has a nearly 80% chance of approaching 0.5, whereas the chance decreases to about 70% when  $\delta = 0.2$

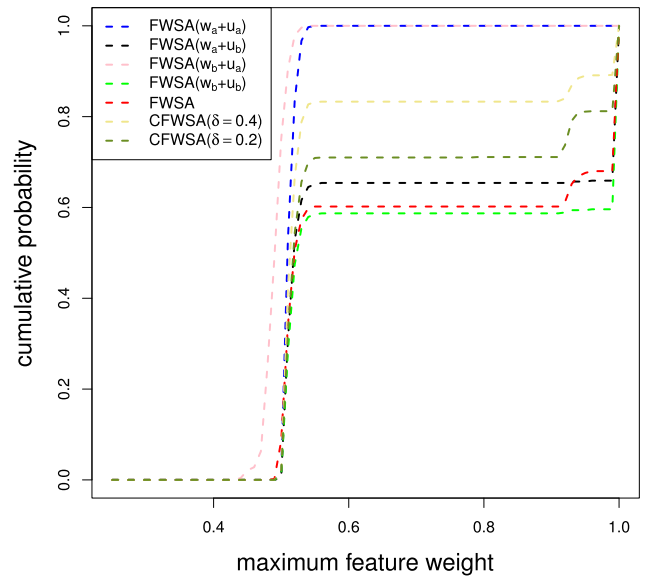


FIGURE 5. Cumulative distributions of the maximum feature weight using different initialization approaches.

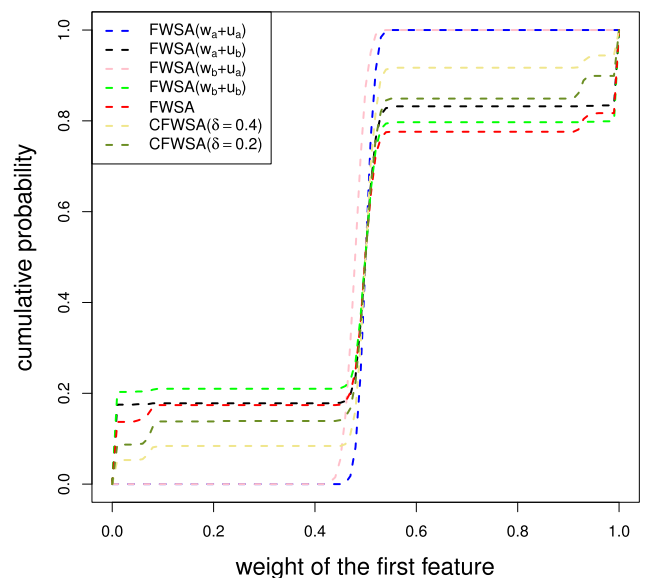


FIGURE 6. Cumulative distributions of the weight of the first feature using different initialization approaches.

(State G). Moreover, the CFWSA has 90% chance of avoiding the one-dimensional solution when  $\delta = 0.4$ , whereas the chance becomes 80% when  $\delta = 0.2$  (Figure 5).

This experiment suggests that initial cluster centers are more important than initial feature weight. This is contrary to the judgement in III B, which tells that they are equally important. There are two reasons. Firstly, the dataset used in III B has two stable partitions whose difference is not big, whereas the dataset used here has only one clear stable partition. Therefore, it is harder to change the partition when true cluster centers are presented. Secondly, and more



importantly, only equal feature weight has been tried here, so more different feature weights should be tested.

### 3) EXPERIMENT ON THE RELATION BETWEEN INITIAL CLUSTER CENTERS AND INITIAL FEATURE WEIGHTS

The result of the last experiment suggests initial cluster centers may be more important. This experiment tests whether we are to get good result when true initial cluster centers are provided. Here we run FWSA K-Means, and initial cluster center set is  $u_a$ . Three additional feature weights are tested:  $w_c = (0.5, 0, 0.5, 0)$ ,  $w_d = (0, 0, 0.5, 0.5)$  and  $w_e = (0.05, 0.05, 0.45, 0.45)$ .

The cumulative distributions of the weight of the first feature are presented in figure 7. And the cumulative distributions of the maximum feature weight are shown in figure 8. The curve of last experiments' State A (the blue line) is kept for comparison. Clearly, 3 different initial feature weights lead to significantly different results. Both  $w_c$  and  $w_d$  very likely lead the algorithm to the one-dimensional solution. On the other hand, though  $w_e$  is also far away from the true weight, it often leads to the true feature weight and completely avoids the one-dimensional solution.

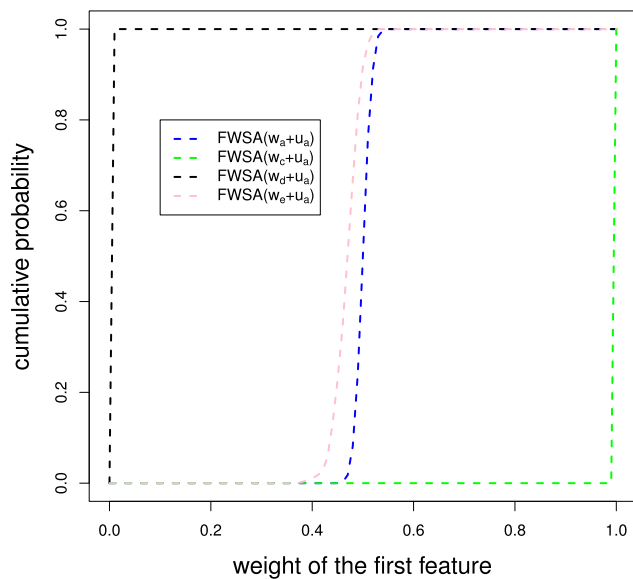


FIGURE 7. Cumulative distributions of the weight of the first feature using different initial feature weights.

We further check the cluster centers after first round of iteration. When feature weight is  $w_c$  or  $w_d$ , the cluster centers change a lot in the first two dimensions, whereas  $w_e$  leads only a minor change to cluster centers. In figure 9, we show the distribution of the closest cluster center according to point (1,1) in the first 2 dimensions. When feature weight is  $w_c$  or  $w_d$ , the cluster center goes to the boundary areas between clusters. With such cluster centers, even traditional K-Means will return unpredictable partitions [6].

From the last 2 experiments we can see that both initial cluster centers and initial feature weights are important, because each of them can exert enough influence to change

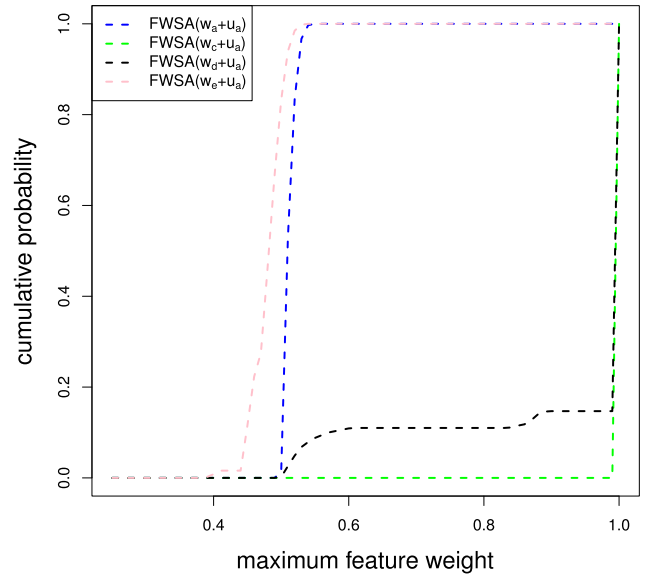


FIGURE 8. Cumulative distributions of the maximum feature weight using different feature weights.

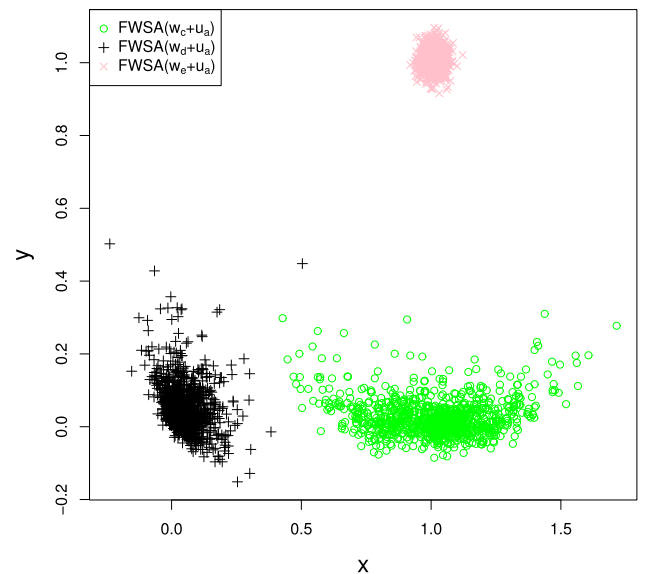


FIGURE 9. Distributions of the cluster center which is the closest to point (1,1) after first iteration.

the other one significantly. As a result, WKM approaches should initialize both of them carefully.

### C. EVALUATION ON REAL DATASETS

#### 1) DATASETS AND ALGORITHMS

We test CFWSA K-Means on 12 UCI datasets, which have also been used for evaluating MAP K-Means [35], [36]. All data are preprocessed with z-score normalization. A brief introduction of the datasets is in Table 1. Ionosphere's 2nd feature has been removed. The abbreviation of a dataset is presented in the parentheses after the dataset's name (for Table 2, 3 and 4).

**TABLE 1. Descriptions of the data sets.**

Dataset	Size	feature	Class
Iris	150	4	3
Wine	178	13	3
Abalone(Aba)	4177	8	3
WDBC	569	30	2
Glass	214	9	6
Ionosphere(Ionos)	351	33	2
Optdigits(Optd)	5620	62	10
Pendigits(Pend)	10992	16	10
Pgblocks(Pgb)	5473	10	5
Pima	768	8	2
Vowel	990	10	11
Waveform(Wave)	5000	21	3

In addition to FWSA K-Means and CFWSA K-Means, 2 feature level constrains based WKM algorithms, which are CFP and MAP K-Means, are compared for evaluating the effectiveness of constraints. Moreover, two initialization approaches for K-Means, which are K-Means++ and BRIk, are compared for evaluating the effectiveness of initialization. In addition, we also combined BRIk into CFWSA K-Means, using it to start the initialization of cluster centers. So totally 7 approaches are compared:

(1) FWSA K-Means (abbreviated as FKM in Table 2, 3 and 4): Algorithm 1 (introduced in section III A).

(2) Constrained FWSA K-Means (abbreviated as CFK in Table 2, 3 and 4): Algorithm 2 (proposed in section IV C).

(3) CFP: proposed in [33], default parameter values are used.

(4) MAP K-Means (abbreviated as MKM in Table 2 and 3): proposed in [35], default parameter values are used.

(5) FWSA K-Means + BRIk (abbreviated as FKB in Table 2 and 3): Algorithm 1, but using BRIk to initialize cluster centers at step 1 (BRIk is proposed in [15]).

(6) FWSA K-Means + K-Means++ (abbreviated as FK+ in Table 2 and 3): Algorithm 1, but using K-Means++ to initialize cluster centers at step 1 (K-Means++ is proposed in [12]).

(7) Constrained FWSA K-Means + BRIk (abbreviated as CFB in Table 2, 3 and 4): Algorithms 2, but using BRIk to start cluster centers at step 6.

Some algorithms' codes are kindly contributed by others:

(1) Both CFP and MAP K-Means codes are updated from <https://github.com/adnan-el-moussawi/MAPK-Means> [35].

(2) The K-Means++ codes are from <https://github.com/jacquesattack/kmeanspp>.

(3) Algorithm BRIk is provided by function *brik* from an R package *brikmeans* [15].

## 2) QUALITY MEASURE

Two quality criteria, which are rand index and normalized mutual information(NMI) are used to evaluate partitions. Ten times of 10-fold cross validation are performed.

## 3) CLUSTERING WITH SUFFICIENT CONSTRAINTS

Firstly,  $p$  randomly generated constraints are added into the clustering process of CFP, Constrained FWSA K-Means (CFK), Constrained FWSA K-Means started with BRIk (CFB). Here  $p$  is the number of data dimension. On the other hand, true feature weight is provided to MAP K-Means(MKM) as the preference vector.

**TABLE 2. Comparison of the rand index averages of the partitions obtained using different approaches.**

Dataset	FKM	CFK	CFP	MKM	FKB	FK+	CFB
Iris	89.32	94.03	90.94	84.66	93.62	90.77	<b>95.02</b>
Wine	86.48	89.09	89.83	<b>92.84</b>	90.13	86.47	90.59
Aba	61.08	61.11	61.03	59.45	61.11	61.06	<b>61.13</b>
WDBC	82.21	83.44	82.53	<b>86.65</b>	82.61	82.01	83.44
Glass	66.73	66.93	<b>68.17</b>	57.91	64.98	65.66	66.93
Ionos	59.2	<b>59.24</b>	53.95	58.93	58.48	59.04	58.69
Optd	88.71	<b>89.5</b>	85.21	74.74	82.91	86.63	84.64
Pend	91.23	91.27	86.18	90.76	<b>91.83</b>	91.33	91.78
Pgb	44.98	46.87	53.56	<b>79.09</b>	64.66	57.26	66.97
Pima	54.64	56.27	55.02	<b>62.46</b>	54.08	54.28	57.17
Vowel	85.77	86.36	85.32	<b>86.44</b>	85.57	85.72	86.2
Wave	66.73	66.73	64.57	<b>66.74</b>	<b>66.74</b>	<b>66.74</b>	<b>66.74</b>
CFK win	11	—	9	6	7	9	3
CFB win	10	7	10	5	10	9	—

**TABLE 3. Comparison of the NMI averages of the partitions obtained using different approaches.**

Dataset	FKM	CFK	CFP	MKM	FKB	FK+	CFB
Iris	83.86	89.4	85.33	75.46	88.46	84.35	<b>90.81</b>
Wine	76.18	79.74	81.71	<b>87.22</b>	81.28	75.49	82.23
Aba	16.3	16.4	16.47	<b>17.47</b>	16.39	16.31	16.41
WDBC	59.6	61.87	57.76	<b>66.35</b>	60.38	59.23	61.44
Glass	46.32	47.66	<b>49.83</b>	46.88	48.31	45.83	49.34
Ionos	16.84	<b>16.93</b>	7.25	16	14.98	16.26	15.49
Optd	59.48	<b>62.15</b>	39.57	48.2	47.98	55.75	52.59
Pend	68.08	68.5	43.73	<b>69.29</b>	68.93	68.32	68.85
Pgb	12.64	13.19	<b>15.32</b>	7.88	13.91	10.67	15.15
Pima	5.23	7.81	7.54	<b>16.38</b>	8.79	4.74	8.79
Vowel	47.65	50.03	43.1	<b>50.61</b>	47.74	47.57	49.58
Wave	36.2	36.23	25.66	<b>36.63</b>	36.3	36.23	36.28
CFK win	12	—	8	5	6	11	4
CFB win	10	7	9	4	9	10	—

The performance of the 7 algorithms are compared in table 2 and 3.

(1) Feature level constraints clearly improves FWSA K-Means. When we compare the third column (CFK, Constrained FWSA K-Means) with the second column (FKM, FWSA K-Means), it wins 11 times in the 12 experiments according to rand index, and wins all according to NMI. In addition, Constrained FWSA K-Means also wins two unsupervised initialization approach, which are FWSA K-Means with K-Means++(FK+) and FWSA K-Means with BRIk (FKB). Moreover, Constrained FWSA K-Means wins CFP, a feature level based WKM. And it is only a little inferior to MAP K-Means, which is provided with the whole true feature weight vector. This also suggests initialization is very important for WKM.

(2) We may improve Constrained FWSA K-Means by using initialization methods for K-Means to start the initialization of cluster centers. We have tried to incorporate BRIk (at step 6 of Algorithm 2), sometimes there is a significant improvement (dataset PageBlocks).

4) CLUSTERING WITH LIMITED AND NOISY CONSTRAINTS

Then we test algorithms using only  $p/2$  noisy constraints, As MAP K-Means requires the whole feature weight vector, it is not included. In addition, most unsupervised algorithms are not included. We keep only FWSA K-Means (FKM) for comparison. The comparison of the NMI averages is shown in table 4. The result of rand index is similar, so we do not present them here.

TABLE 4. Comparison of the NMI averages of the partitions obtained using different approaches with limited and noisy constraints.

Dataset	FKM	CFK	CFP	CFB
Iris	83.86	84.07	74.73	<b>90.22</b>
Wine	76.18	80.49	76.66	<b>82.22</b>
Aba	16.3	16.33	16.21	<b>16.41</b>
WDBC	59.6	60.33	52.38	<b>60.48</b>
Glass	46.32	47.14	46.14	<b>47.93</b>
Ionos	16.84	<b>17.56</b>	8.25	15.87
Optd	59.48	<b>60.27</b>	37.39	51.86
Pend	68.08	68.52	41.46	<b>69.15</b>
Pgb	12.64	12.91	<b>15.15</b>	14.81
Pima	5.23	5.18	<b>7.71</b>	6.17
Vowel	47.65	<b>49.54</b>	43.44	48.62
Wave	36.2	36.2	25.38	<b>36.24</b>
CFK win	10	—	10	3
CFB win	10	9	10	—

Compared with table 3, we can find using limited and noisy constraints instead of enough and precise ones causes only a minor decrease in the performance of Constrained FWSA K-Means (CFK) and Constraint FWSA K-Means started with BRIk (CFB), whereas it leads a significant decrease for CFP. This is because we use constraints only at the stage of initialization. As the noisy constraints are not against the order of the true feature weights, they are helpful in getting a good initial state. The following clustering process does not consider the constraints, so the final partition is more robust to the noise in constraints.

5) CHOICE OF THE PARAMETERS

Constrained FWSA K-Means uses two parameters: rounds of adjusting initial feature weight  $N_w$ , and rounds of adjusting initial cluster centers  $N_c$ . In our experiments, we set  $N_w = 20$  and  $N_c = 10$ . Based upon Iris and Wine dataset, we test the impact of changing parameters in two scenarios: which are  $p$  precise constraints, and  $p/2$  noisy constraints. The results are shown in figure 10 and 11.

Figure 10 suggests that the clustering quality improves with the increase of  $N_c$  when  $N_c$  is less than 10. After that, the changes are no longer significant. So we set  $N_c = 10$  in all experiments.

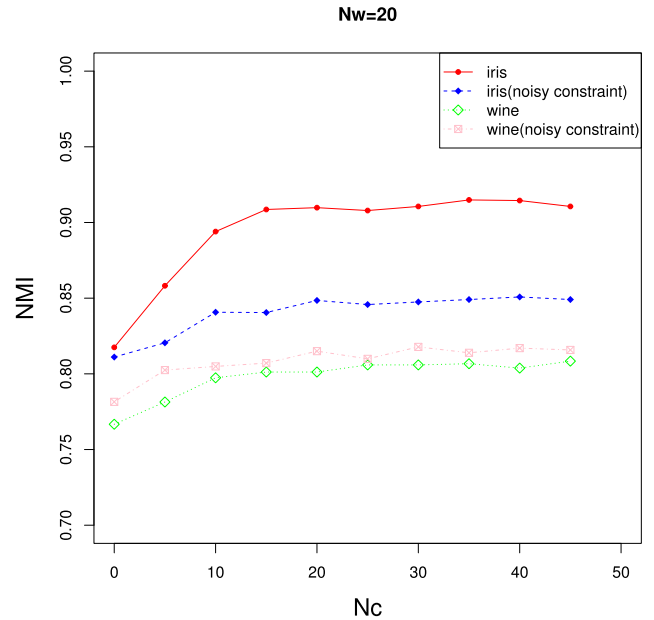


FIGURE 10. Influence of the rounds of adjusting initial cluster centers  $N_c$ .

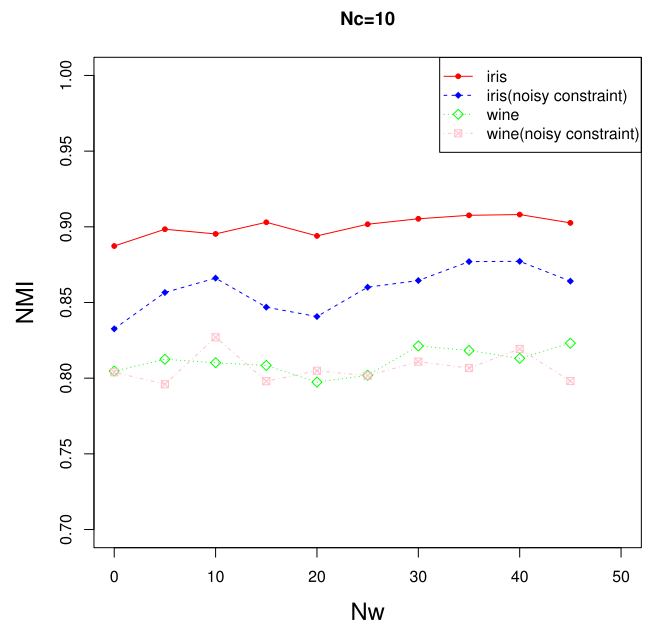


FIGURE 11. Influence of the rounds of adjusting initial feature weight  $N_w$ .

Figure 11 suggests that parameter  $N_w$ 's influence is not as significant as  $N_c$  does. This is because the feature weight continues its evolution in the following stage with the evolution of clustering centers. As  $N_w = 0$  often causes a relative worse result, we may set it as a positive number which is not too small. As the computation cost for adjusting initial feature weight is marginal, we set  $N_w = 20$  in all experiments.

VI. CONCLUSION

As a K-Means type algorithm, WKM often requires initial cluster centers. In addition, it uses an initial feature weight

vector. As a result, WKM's initialization problem is more complicated than that of K-Means. Yet, the problem is rarely addressed. This paper analyzes experimentally the initialization of FWSA K-Means, a popular WKM method proposed to avoid the one-dimensional solution. It suggests that FWSA K-Means must initialize carefully to sidestep the solution. Then it proposes a novel semi-supervised initialization approach, which utilizes feature level constraints to guide the evolution of feature weights and cluster centers. Experimental results prove the effectiveness of feature level constraints to the initialization.

This paper analyses the initialization of WKM experimentally. We do not intend to analyse the problem theoretically here. This is our future work. On the other hand, as initialization is the focus of this research, feature level constraints have not been applied to improve the iterative process of FWSA K-Means. This is also our future research direction.

## REFERENCES

- [1] E. Hancer, B. Xue, and M. Zhang, "A survey on feature selection approaches for clustering," *Artif. Intell. Rev.*, vol. 53, no. 6, pp. 4519–4545, Aug. 2020.
- [2] D. S. Modha and W. S. Spangler, "Feature weighting in  $K$ -means clustering," *Mach. Learn.*, vol. 52, no. 3, pp. 217–237, Sep. 2003.
- [3] E. Y. Chan, W. K. Ching, M. K. Ng, and J. Z. Huang, "An optimization algorithm for clustering using weighted dissimilarity measures," *Pattern Recognit.*, vol. 37, no. 5, pp. 943–952, May 2004.
- [4] Z. Deng, K. Choi, Y. Jiang, J. Wang, and S. Wang, "A survey on soft subspace clustering," *Inf. Sci.*, vol. 348, pp. 84–106, Jun. 2016.
- [5] R. C. de Amorim, "A survey on feature weighting based  $K$ -means algorithms," *J. Classification*, vol. 33, no. 2, pp. 210–242, Jul. 2016.
- [6] S. Bubeck, M. Meilă, and U. von Luxburg, "How the initialization affects the stability of the  $k$ -means algorithm," *ESAIM Probab. Statist.*, vol. 16, pp. 436–452, Sep. 2012.
- [7] H. Frigui and O. Nasraoui, "Unsupervised learning of prototypes and attribute weights," *Pattern Recognit.*, vol. 37, no. 3, pp. 567–581, Mar. 2004.
- [8] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the  $k$ -means clustering algorithm," *Expert Syst. Appl.*, vol. 40, no. 1, pp. 200–210, Jan. 2013.
- [9] P. Fränti and S. Sieranoja, "How much can  $k$ -means be improved by using better initialization and repeats?" *Pattern Recognit.*, vol. 93, pp. 95–112, Sep. 2019.
- [10] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theor. Comput. Sci.*, vol. 38, pp. 293–306, Jan. 1985.
- [11] I. Katsavounidis, C.-C. J. Kuo, and Z. Zhang, "A new initialization technique for generalized Lloyd iteration," *IEEE Signal Process. Lett.*, vol. 1, no. 10, pp. 144–146, Oct. 1994.
- [12] D. Arthur and S. Vassilvskii, " $k$ -means++: The advantages of careful seeding," in *Proc. Annu. ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2007, pp. 1027–1035.
- [13] T. Onoda, M. Sakai, and S. Yamada, "Careful seeding method based on independent components analysis for  $k$ -means clustering," *J. Emerg. Technol. Web Intell.*, vol. 4, no. 1, pp. 51–59, Feb. 2012.
- [14] T. Su and J. G. Dy, "In search of deterministic methods for initializing  $K$ -means and Gaussian mixture clustering," *Intell. Data Anal.*, vol. 11, no. 4, pp. 319–338, Jul. 2007.
- [15] A. Torrente and J. Romo, "Initializing  $k$ -means clustering by bootstrap and data depth," *J. Classification*, vol. 38, pp. 232–256, Jul. 2021.
- [16] P. Fränti and J. Kivijärvi, "Randomised local search algorithm for the clustering problem," *Pattern Anal. Appl.*, vol. 3, no. 4, pp. 358–369, Dec. 2000.
- [17] M. Rezaei, "Improving a centroid-based clustering by using suitable centroids from another clustering," *J. Classification*, vol. 37, pp. 352–365, Jul. 2020.
- [18] M. C. Naldi, R. J. G. B. Campello, E. R. Hruschka, and A. C. P. L. F. Carvalho, "Efficiency issues of evolutionary  $k$ -means," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1938–1952, Mar. 2011.
- [19] Z. He and C. Yu, "Clustering stability-based evolutionary  $K$ -means," *Soft Comput.*, vol. 23, no. 1, pp. 305–321, Jan. 2019.
- [20] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li, "Automated variable weighting in  $k$ -means type clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 657–668, May 2005.
- [21] C.-Y. Tsai and C.-C. Chiu, "Developing a feature weight self-adjustment mechanism for a  $K$ -means clustering algorithm," *Comput. Statist. Data Anal.*, vol. 52, no. 10, pp. 4658–4672, Jun. 2008.
- [22] R. C. De Amorim and B. Mirkin, "Minkowski metric, feature weighting and anomalous cluster initializing in  $K$ -means clustering," *Pattern Recognit.*, vol. 45, no. 3, pp. 1061–1075, Mar. 2012.
- [23] R. C. de Amorim and P. Komisarczuk, "On initializations for the Minkowski weighted  $k$ -means," in *Proc. 11th Int. Conf. Adv. Intell. Data Anal.*, 2012, pp. 45–55.
- [24] R. Deepana, "On sample weighted clustering algorithm using Euclidean and Mahalanobis distances," *Int. J. Statist. Syst.*, vol. 12, no. 3, pp. 421–430, Jul. 2017.
- [25] S. Basu, A. Banerjee, and R. J. Mooney, "Semi-supervised clustering by seeding," in *Proc. 19th Int. Conf. Mach. Learn. (ICML)*, 2002, pp. 27–34.
- [26] V.-V. Vu, N. Labroche, and B. Bouchon-Meunier, "Active learning for semi-supervised  $K$ -means clustering," in *Proc. 22nd IEEE Int. Conf. Tools Artif. Intell. (ICTAI)*, Oct. 2010, pp. 12–15.
- [27] J. Yoder and C. E. Priebe, "Semi-supervised  $k$ -means++," *J. Stat. Comput. Simul.*, vol. 87, no. 13, pp. 2597–2608, Jun. 2017.
- [28] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, pp. 81–88.
- [29] A. Vouros, S. Langdell, M. Croucher, and E. Vasilaki, "An empirical comparison between stochastic and deterministic centroid initialisation for  $K$ -means variations," *Mach. Learn.*, vol. 110, no. 8, pp. 1975–2003, Aug. 2021.
- [30] G. Guo, S. Chen, and L. Chen, "Soft subspace clustering with an improved feature weight self-adjustment mechanism," *Int. J. Mach. Learn. Cybern.*, vol. 3, no. 1, pp. 39–49, Mar. 2012.
- [31] L. Jing, M. K. Ng, and J. Z. Huang, "An entropy weighting  $k$ -means algorithm for subspace clustering of high-dimensional sparse data," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 8, pp. 1026–1041, Aug. 2007.
- [32] P. Zhou, J. Chen, M. Fan, L. Du, Y.-D. Shen, and X. Li, "Unsupervised feature selection for balanced clustering," *Knowl.-Based Syst.*, vol. 193, Apr. 2020, Art. no. 105417.
- [33] J. Sun, W. Zhao, J. Xue, Z. Shen, and Y. Shen, "Clustering with feature order preferences," in *Proc. 10th Pacific Rim Int. Conf. Artif. Intell. (PRICAI)*, 2008, pp. 382–393.
- [34] J. Wang, S. Wu, and G. Li, "Clustering with instance and attribute level side information," *Int. J. Comput. Intell. Syst.*, vol. 3, no. 6, pp. 770–785, Dec. 2010.
- [35] A. El Moussawi, A. Giacometti, N. Labroche, and A. Soulet, "MAPK-means: A clustering algorithm with quantitative preferences on attributes," *Intell. Data Anal.*, vol. 24, no. 2, pp. 459–489, Mar. 2020.
- [36] D. Dua and C. Graff, *UCI Machine Learning Repository*. Irvine, CA, USA: Univ. of California, 2019. [Online]. Available: <http://archive.ics.uci.edu/ml>



**ZHENFENG HE** received the Ph.D. degree in control engineering from the University of Science and Technology of China, in 2004. He is currently an Associate Professor with the College of Computer and Data Science, Fuzhou University, China. His research papers have been published in soft computing, knowledge-based systems, and other professional journals. His research interests include machine learning, data mining, and their applications in agriculture.

•••