Journal of Statistical Theory and Applications
Vol. **18(2)**, June 2019, *pp.* 147–154
DOI: https://doi.org/10.2991/jsta.d.190524.001; ISSN 1538-7887
https://www.atlantis-press.com/journals/jsta

The LINEX Weighted k-Means Clustering

Narges Ahmadzadehgoli[1], Adel Mohammadpour[2,*], Mohammad Hassan Behzadi[1]

[1]*Department of Statistics, Science and Research Branch, Islamic Azad University, Tehran, Iran*
[2]*Department of Statistics, Faculty of Mathematics & Computer Science, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Avenue, Tehran, Iran*

ARTICLE INFO

Article History
Received 24 Aug 2017
Accepted 30 June 2018

Keywords
LINEX loss function
Feature weights
Weighted k-means
Clustering

ABSTRACT

LINEX weighted k-means is a version of weighted k-means clustering, which computes the weights of features in each cluster automatically. Determining which entity is belonged to which cluster depends on the cluster centers. In this study, the asymmetric LINEX loss function is used to compute the dissimilarity in the weighted k-means clustering. So, the cluster centroids are obtained by minimizing a LINEX based cost function. This loss function is used as a dissimilarity measure in clustering when one wants to overestimate or underestimate the cluster centroids, which helps to reduce some errors of misclassifying entities. Therefore, we discuss the LINEX weighted k-means algorithm. We examine the accuracy of the algorithm with some synthetic and real datasets.

1. INTRODUCTION

Consider the dataset $\boldsymbol{X} = (\boldsymbol{X}_1, \dots, \boldsymbol{X}_n)'$ with n entities and m features $\boldsymbol{X}_i = (X_{i1}, \dots, X_{im})$. The aim of clustering is to group the observations of a dataset into K disjoint similar clusters, $\boldsymbol{S} = \{S_1, \dots, S_K\}$. The most common clustering algorithm is k-means [1] which partitions the dataset in different groups by minimizing the dissimilarity between objects and their corresponding cluster centroids $\boldsymbol{C}_k \in (\boldsymbol{C}_1, \dots, \boldsymbol{C}_K)'$, $\boldsymbol{C}_k = (C_{k1}, \dots, C_{km})$, for $k = 1, \dots, K$. Consider the following cost function:

$$J(\boldsymbol{H}, \boldsymbol{C}) = \sum_{k=1}^{K} \sum_{i=1}^{n} h_{ik} L(\boldsymbol{X}_i, \boldsymbol{C}_k),$$

where $\sum_{k=1}^{K} h_{ik} = 1$ and $h_{ik} \in \{0, 1\}$ is a variable that represents weather if $\boldsymbol{X}_i$ belongs to S_k or not. L can be a famous symmetric measure such as Euclidean, City Block, and Minkowski. In k-means clustering algorithms, all features in a dataset have an equal effect on results. However, in fact, all the features are not equally importance and some of them are noisy. Huang *et al.* [2] have proposed weighted k-means clustering algorithm (Wk-means) which assigned different weights to each feature. It minimized the below equation

$$J_{\beta,2}(\boldsymbol{H}, \boldsymbol{C}, \boldsymbol{W}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{d=1}^{m} h_{ik} w_d^{\beta} |\boldsymbol{X}_{id} - \boldsymbol{C}_{kd}|^2, \tag{1}$$

where $\sum_{k=1}^{K} h_{ik} = 1$ and $\boldsymbol{H}$ is a matrix that includes $h_{ik} \in \{0, 1\}$, $\boldsymbol{W} = (w_1, \dots, w_m)$ is the vector of feature weights, each w is nonnegative such that $\sum_{d=1}^{m} w_d = 1$ and $\beta \geq 1$ represents the impact of weights (when $\beta < 0$, it assigns small weights to the features, therefore in this work we consider $\beta \geq 1$.). To minimize the cost function (1) between X_i and its corresponding centroid C_k in each class subject to $\sum_{d=1}^{m} w_d = 1$ ($w_d \geq 0$), the weights are updated using the following relation:

$$w_d = \begin{cases} 0 & E_d = 0 \\ \dfrac{1}{\sum_{u \in M} \left[\frac{E_d}{E_u}\right]^{\frac{1}{\beta-1}}} & E_d \neq 0, M = \{1, \dots, m\}, \end{cases} \tag{2}$$

where

$$E_d = \sum_{k=1}^{K} \sum_{i=1}^{n} h_{ik} |X_{id} - C_{kd}|^2.$$

*Corresponding author. Email: adel@aut.ac.ir

The Wk-mean algorithm is as below:

1. Specify the number of clusters, K. Set the initial centroids and feature weights randomly such that $\sum_{d=1}^{m} w_d = 1$, it could be $w_d = \frac{1}{m}$.
2. Allocate each entity to the nearest centroid by minimizing (1).
3. All centroids must be updated to the centers of the corresponding groups. Stop if there is no change in clusters of step 2. Then, $\boldsymbol{S} = \{S_1, \ldots, S_K\}$ are the final partitions.
4. Using (2), the weights are updated by considering the constraint $\sum_{d=1}^{m} w_d = 1$

Amorin *et al.* [3] generalized this algorithm to the Minkowski Wk-means (MWk-means), which used the Minkowski distance as the dissimilarity measure instead of Euclidean. It minimizes the following criterion:

$$J_{\beta,\beta}(\boldsymbol{H}, \boldsymbol{C}, \boldsymbol{W}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{d=1}^{m} h_{ik} w_d^{\beta} |X_{id} - C_{kd}|^{\beta}.$$

The MWk-means algorithm is used to classify the dataset $\boldsymbol{X}_{n\times m}$ into K disjoint clusters of analogous objects using the Minkowski distance such that different weights are assigned to each feature at different clusters. Choosing the dissimilarity measure in clustering algorithms is an important issue. Modha and Spangler [4] explained that one could use a nonnegative, complex, and symmetric loss function as the dissimilarity measure in the clustering, and they introduced the convex k-means algorithm. Kummamuru *et al.* [5] illustrated the class of dissimilarity measures, which are asymmetric. Parsian and Kirmani [6] explained that the symmetric loss functions are useful when the overestimating and underestimating of the estimator are not important; otherwise, an asymmetric loss function as LINEX is appropriate. For example, the risk of overfilling the gasoline tank of an aircraft is less than underfilling it, or misdiagnosis of cancer in a patient might be more costly. LINEX is an asymmetric loss function, which is convex and has some interesting properties. For example, it is exponential or linear for different input values according to its parameter. In the next part, we talk more about this loss function, see [6].

Here, we want to generalize the Wk-means algorithms to the one with the asymmetric LINEX loss function, as the dissimilarity measure. We call this LINEX weighted k-means (LINEX Wk-means) clustering algorithm. It is useful, especially when one wants to force some data to place in specific clusters to reduce some losses, while it assigns different weights to each feature. For example, misclassifying a set of bombs into two groups of high energy or low energy may lead to hazardous results. To check the accuracy of our LINEX Wk-means algorithm, which means how much it can partition the data well, we use some synthetic and real datasets, which are labeled before. We also compute the Davies–Bouldin (DB) index, the normalized variation information (NVI), and an accuracy measure (AM) that measures the percent of actual clustering labels to compare the results.

2. LINEX k-MEANS ALGORITHMS

Sometimes we deal with clustering a dataset such that misclassification of an entity into a specific cluster might be costly, hazardous, or we tend to put some data into a particular cluster. In this case, the k-means algorithm with a symmetric dissimilarity measure may not be so appropriate. Therefore, we need to use an asymmetric dissimilarity measure like LINEX so that the overestimating or underestimating error is different. This algorithm is called LINEX k-means algorithm [7].

Suppose the parameter $\theta \in \Theta$ be unknown and let $\delta(X)$ be an estimator of θ which is based on X. Then $\Lambda = \delta(X) - \theta$ is the error of estimating θ. The LINEX loss function, which was proposed by Varian [8], is as the following:

$$L_{\text{LINEX}} = (\Lambda) = \exp(a\Lambda) - (a\Lambda) - 1, \tag{3}$$

where $a \in R$ is a scalar and $L_{\text{LINEX}}(\Lambda)$ is convex, asymmetric, and nonnegative. $L_{\text{LINEX}}(\Lambda)$ is linear for negative Λ if $a < 0$ and it is exponential if $a > 0$ for positive Λ. For small values of a, $L_{\text{LINEX}}(\Delta)$ is not far from the symmetric Euclidean loss. So the optimal estimate falls on the mean which is obtained by the Euclidean one. This means that estimation under LINEX loss is consistent with one achieved from the Euclidean loss (when $a \to 0$). When the overestimating is more costly than the underestimating, $L_{\text{LINEX}}(\Lambda)$ is completely asymmetric and a is close to one [6]. The multiparameter case is as below,

$$L_{\text{LINEX}}(\Lambda) = \sum_{d=1}^{m} \left(\exp(a_d\Lambda_d) - a_d\Lambda_d - 1\right),$$

where $a_d \neq 0$, $\Lambda_d = \delta_d(X) - \theta_d$ and $\Lambda = (\Lambda_1, \ldots, \Lambda_m)$.

Now we want to use the LINEX loss function in the weighted k-means clustering, as the dissimilarity measure. Suppose we have a dataset $\boldsymbol{X}$, as was introduced before. The LINEX k-means algorithm uses the following cost function to measure the distance between each entity

in a cluster and its corresponding centroid,

$$J_{\text{LINEX}}(\boldsymbol{H}, \boldsymbol{C}, a) = \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{d=1}^{m} h_{ik}\left[\exp\left(a\left(X_{id} - C_{kd}\right)\right) - a\left(X_{id} - C_{kd}\right) - 1\right].$$

The process is like the k-means algorithm except in the dissimilarity measure and optimal points of centroids in each cluster. The optimal centers are achieved by minimizing $J_{\text{LINEX}}(\boldsymbol{H}, \boldsymbol{C}, a)$ with respect to $\boldsymbol{C}$, [7].

$$C_{kj} = \frac{1}{a}\log\frac{\sum_{i=1}^{n} h_{ik}e^{aX_{id}}}{\sum_{i=1}^{n} h_{ik}}, \text{ for } d = 1, \ldots, m.$$

When a is close to zero, the results are close to the common k-means algorithm with symmetric dissimilarity measures. When $a > 0$, the overestimation is more important than the underestimation and the entities which are close to the border of two clusters are pushed into a cluster, according to the overestimated centers.

For large values of X_{id}, since $e^{X_{id}}$ is not computable, so C_{kj} cannot be evaluated. Therefore, before processing, the data should be standardized by the following relation:

$$Z_{id} = \frac{X_{id} - \min_d\left(X_{id}\right)}{\max_d\left(X_{id}\right) - \min_d\left(X_{id}\right)}.$$

When a is small enough, $e^{aX_{id}}$ is computable and normalizing the data is not needed.

3. THE LINEX Wk-MEANS CLUSTERING

LINEX Wk-means clustering is a generalization of Wk-means and LINEX k-means algorithms. It assigns different weights to each feature while using a LINEX loss function as the dissimilarity measure. The processes are as the Wk-means algorithm with some differences. Consider the following equation:

$$J_1(\boldsymbol{H}, \boldsymbol{C}, \boldsymbol{W}, a) = \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{d=1}^{m} h_{ik}w_d^{\beta}\left[\exp\left(a\left(X_{id} - C_{kd}\right)\right) - a\left(X_{id} - C_{kd}\right) - 1\right], \tag{4}$$

where each weight is nonnegative, $\sum_{d=1}^{m} w_d = 1$ and β is the impact of weights and $h_{ik} \in \{0, 1\}$, as it was defined before. The goal is to find the optimal centers and feature weights $w_d^{*(1)}$, for $d = 1, \ldots, m$, by minimizing (4). We rewrite (4) as

$$J_1(\boldsymbol{H}, \boldsymbol{C}, \boldsymbol{W}, a) = \sum_{d=1}^{m} w_d^{\beta}E_d, \tag{5}$$

where

$$E_d = \sum_{k=1}^{K}\sum_{i=1}^{n} h_{ik}\left[\exp\left(a\left(X_{id} - C_{kd}\right)\right) - a\left(X_{id} - C_{kd}\right) - 1\right].$$

First, we find the optimal feature weights according to the constraint $\sum_{d=1}^{m} w_d = 1$, [3,9]. We derive the Lagrange function $L_1 = \sum_{d=1}^{m} w_d^{\beta}E_d + \lambda\left(1 - \sum_{d=1}^{m} w_d\right)$ with respect to w_d. Therefore,

$$\frac{\partial L_1}{\partial W_d} = \beta w_d^{\beta-1}E_d - \lambda.$$

Equating it to zero leads to $w_d = \left(\frac{\lambda}{\beta E_d}\right)^{\frac{1}{\beta-1}}$ and by summing over d, $1 = \sum_{d=1}^{m}\left(\frac{\lambda}{\beta E_d}\right)^{\frac{1}{\beta-1}}$ and finally,

$$w_d^{*(1)} = \begin{cases} 0 & E_d = 0 \\ \dfrac{1}{\sum_{u\in M}\left[\frac{E_d}{E_u}\right]^{\frac{1}{\beta-1}}} & E_d \neq 0, M = \{1, \ldots, m\}. \end{cases} \tag{6}$$

When $E_d = 0$, a unique value is assigned to the d-th variable in each cluster. So, we put $w_d^{*(1)} = 0$. If E_u is equal to zero for a $u \in M$ then (6) is not efficient. To avoid these conditions, adding a positive constant to E_d is suggested [9]. This constant could be the average of the dispersion of the features in a dataset. When $\beta = 1$, the minimum of (4) is then achieved at $w_{d^*}^{*(1)} = 1$ and the weights of the other features are zero, where d^* is the feature that has the smallest sum of within-cluster variance. The optimum weight depends on the ratio of $\left[\frac{E_d}{E_u}\right]$, which are based on the LINEX loss directly. It seems overestimation or underestimation affect just on centers (not weights).

Now it is turned to find the optimal centers. Consider the following optimization problem:

$$T(\boldsymbol{H}, \boldsymbol{C}, \boldsymbol{W}) = \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{d=1}^{m} h_{ik} w_d^{\beta} L_{LINEX}(\boldsymbol{X}_i, \boldsymbol{C}_k),$$

where L_{LINEX} is the LINEX dissimilarity measure and we have is $h_{ik} \in \{0, 1\}$ for $i = 1, \ldots, n$ and $k = 1, \ldots, K$, $\sum_{k=1}^{K} h_{ik} = 1$ for $i = 1, \ldots, n$ and $\sum_{d=1}^{m} w_d = 1$. We minimize T($\boldsymbol{H}$, $\boldsymbol{C}$, $\boldsymbol{W}$) according to the above conditions in the following three steps:

1. Fix $\boldsymbol{C} = \widetilde{\boldsymbol{C}}$ and $\boldsymbol{W} = \widetilde{\boldsymbol{W}}$. $T\left(\boldsymbol{H}, \widetilde{\boldsymbol{C}}, \widetilde{\boldsymbol{W}}\right)$ is minimized iff,
   $h_{ik} = 1$ if $L_{LINEX}(\boldsymbol{X}_i, \boldsymbol{C}_k) < L_{LINEX}(\boldsymbol{X}_i, \boldsymbol{C}_r)$ for $1 \le r \le K$,
   $h_{ik} = 0$ if $r \ne k$.
2. Fix $\boldsymbol{C} = \widetilde{\boldsymbol{C}}$ and $\boldsymbol{H} = \widetilde{\boldsymbol{H}}$ and solve $T(\boldsymbol{H}, \boldsymbol{C}, \boldsymbol{W})$. Then T is minimized at $w_d^{*(1)}$ in relation (6).
3. Fix $\boldsymbol{W} = \widetilde{\boldsymbol{W}}$ and $\boldsymbol{H} = \widetilde{\boldsymbol{H}}$ and solve $T(\boldsymbol{H}, \boldsymbol{C}, \boldsymbol{W})$. Then it is minimized if and only if for each entity we have,

$$C_{kd} = \frac{1}{a}\log\frac{\sum_{i=1}^{n} h_{ik} e^{aX_{id}}}{\sum_{i=1}^{n} h_{ik}}, \text{ for } d = 1, \ldots, m. \tag{7}$$

To show step 3, fix k and d, then it is enough to minimize,

$$\sum_{i=1}^{n} h_{ik} w_d^{\beta}\left(\exp\left(a\left(X_{id} - C_{kd}\right)\right) - a\left(X_{id} - C_{kd}\right) - 1\right).$$

By differentiating it with respect to C_{kd} and then equating it to zero,

$$-ae^{-aC_{kd}} w_d^{\beta}\sum_{i=1}^{n} h_{ik} e^{aX_{id}} + aw_d^{\beta}\sum_{i=1}^{n} h_{ik} = 0,$$

and (7) is obtained.

The LINEX Wk-means is the same as the Wk-means algorithm except in cluster centers optimization, weights, and the dissimilarity measure (steps 2, 3, and 4). That is,

2. Allocate each entity to the nearest centroid by minimizing (4).
3. All centroids must be updated to the centers of the corresponding groups using (7).
4. Using (6), the weights are updated by considering the constraint 2.

There is also another version of LINEX Wk-means algorithm, which differs in its exponential weights and the cost function is as the following:

$$J_2(\boldsymbol{H}, \boldsymbol{C}, \boldsymbol{W}, a) = \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{d=1}^{m} h_{ik} e^{w_d}\left[\exp\left(a\left(X_{id} - C_{kd}\right)\right) - a\left(X_{id} - C_{kd}\right) - 1\right], \tag{8}$$

such that $\sum_{d=1}^{m} w_d = 1$. We call it LINEX exponentially weighted k-means algorithm (LINEX EWk-means). The optimal centers are the same as (7), so it is enough to find the optimal feature weights, $w_d^{*(2)}$, by minimizing (8) with respect to $\sum_{d=1}^{m} w_d = 1$. We rewrite (8) as

$$J_2(\boldsymbol{H}, \boldsymbol{C}, \boldsymbol{W}, a) = \sum_{d=1}^{m} e^{w_d} E_d,$$

then we minimize the Lagrange function $L_2 = \sum_{d=1}^{m} e^{w_d} E_d + \lambda \left(1 - \sum_{d=1}^{m} w_d\right)$ with respect to w_d. Therefore,

$$\frac{\partial L_2}{\partial W_d} = e^{w_d} E_d - \lambda.$$

Equating it to zero leads to $w_d = \log\left(\frac{\lambda}{E_d}\right)$ and by summing over d, $1 = \sum_{d=1}^{m} \log\left(\frac{\lambda}{E_d}\right)$. However,

$$\sum_{d=1}^{m} \left(\log(\lambda) - \log(E_d)\right) = m \log(\lambda) - \sum_{d=1}^{m} \log(E_d).$$

Therefore,

$$\log(\lambda) = \frac{1 + \sum_{d=1}^{m} \log(E_d)}{m}$$

and finally

$$w_d^{*(2)} = \begin{cases} 0 & E_d = 0 \\ \dfrac{1 - \sum_{u \in M} \log\left(\frac{E_d}{E_u}\right)}{m}, & E_d \neq 0 \end{cases}. \tag{9}$$

For $u \in M$, we add the average of the dispersion of the features in a dataset to E_u to avoid a division by zero in (9).

4. PERFORMING THE EXPERIMENTS

In comparison with the LINEX k-means clustering, one advantage of the LINEX Wk-means algorithm is to assign different feature weights in a dataset to classify the entities. Here we want to check the performance of LINEX Wk-means on some simulated and real datasets, which are available in the UC Irvine Machine Learning Repository [10]. The simulated datasets are generated from Normal, Log-Normal, Gamma, and Poisson distributions as the following densities:

$$\begin{aligned}
&\text{Normal } (\mu, \sigma): && \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),\ x, \mu \in R, \sigma > 0 \\
&\text{Log-Normal } (\mu, \sigma): && \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log(x) - \mu)}{2\sigma^2}\right),\ x > 0, \sigma > 0, \mu \in R \\
&\text{Gamma } (\alpha, \beta): && \frac{\beta^{\alpha} x^{\alpha-1} e^{-x\beta}}{\Gamma(\alpha)},\ x \geq 0, \alpha, \beta > 0 \\
&\text{Poisson } (\lambda): && \frac{e^{-\lambda} \lambda^{x}}{x!},\ x = 0, 1, \ldots.
\end{aligned}$$

All of these datasets are labeled, which helps us to evaluate the accuracy of an algorithm. To accomplish this, we map these labels to their clusters, which are generated from the algorithm. We run the algorithms several times since the initial centroids are chosen from the entities randomly and lead to different results in each iteration. Each time, the percent of the corrected labels are computed, and finally, the average of the accuracies is calculated. In the next experiment, we add some features to our datasets such that they contain random noises [3,10]. These random noises are generated from uniform density. Then we specify the impact of these noisy features on the algorithm results. We want to represent how much the LINEX Wk-means algorithm results remained fixed when adding these noisy features. Now we introduce the datasets:

4.1. Simulated Datasets

The sum of n independent and identically distributed (iid) random variables with Normal, Gamma (with the same α), and Poisson distributions have the same respective distributions, and the product of some iid Log-Normal random variables is Log-Normal. We produce three datasets with 100 dependent data (using the above rules) and each with three features from Normal, Log-Normal, and Gamma densities. Our fourth dataset is from Poisson density, but with one feature. Each dataset is partitioned into two clusters. The first 50 entities are labeled as 1 and the second 50 entities have labeled 2. To see the procedure of generating these datasets, one can see [7].

4.2. Real Datasets

We choose some real datasets such that classifying some events in a special cluster might be better or worse, since sometimes when some entities are labeled wrongly, they may lead to irreparable losses. Therefore, we say the overestimating and the underestimating are not of equal importance.

Gamma Telescope: This dataset contains 19020 events with 11 features of different high-energy gamma particles. These events are divided into two groups, background, and single. Clustering a background event as a single is worse than its conversing.

Haberman's Survival: It contains 306 data of breast cancer patients with three features. It is divided into two clusters "the patients who were alive five years or more" and "the patients who were alive less than five years."

Seismic bumps: It contains 2584 data with 18 features that represent the energy of seismic bums in a coal mine. This is clustered into two classes "high energy" and "no high energy."

First, we illustrate the performance of the two LINEX Wk-means algorithms for different values of a and β which we choose them in steps of 0.1 in $(0, 1)$ and $[1, 5]$, respectively. Note that when a is close to zero, the results are not so different from Wk-means algorithm with Euclidean distance. Then, we add the noisy features to the datasets to see how much the LINEX Wk-means algorithm is sensitive to these noisy features.

5. EVALUATION

Now we check the results in 14 datasets (8 simulated and 6 real datasets). To evaluate the two versions of LINEX Wk-means algorithms, we compute AM, NVI, and DB [11,12]. NVI and AM are two external criteria that depend on the datasets inherently. To compute them, we need a dataset, which is prelabeled. They compare the output labels of an algorithm with the dataset labels. AM is our accuracy measure that represents the percent of the actual classified observations and is better when it is close to 100. If NVI takes values from 0 to 1 it shows good clustering performance. It decreases, as the clusters are more homogeneous. DB is an internal criterion that depends on the dispersion between clusters and the inherent data. The smaller value of DB states that the clusters are separated well. The computations have been performed on the Intel Core i3 processor CPU 2.13 GHz, Ram 6 GB, on the MATLAB, version 2013a.

5.1. Performance in Simulated Datasets

For each dataset, we run the Wk-means, the LINEX Wk-means, and the LINEX EWk-means 100 times (since the initial centroids are selected from the entities randomly) and obtain the average of AM, NVI, and DB criteria. Table 1 shows the results of the four simulated datasets for the above three algorithms. The parameter β is the power of weights in Wk-means and LINEX Wk-means algorithms and a is the LINEX parameter. The smaller values of AM, NVI, and DB helps to determine better values of a and β. We examine different values of 2 in steps of 0.1. In most datasets, $\beta = 1$ leads to better results in LINEX Wk-means algorithm. As the overestimating and the underestimating in the datasets of Table 1, are not important, we consider a close to zero. In all Tables, the best values of the criteria in each column of different datasets, are bolded.

In Table 2, we add three noisy feature to Normal, Log-Normal, and Gamma datasets and one noisy feature to Poisson dataset to check how much the algorithms are sensitive to the noisy features. As you see the results in Tables 1 and 2 are very close together and in almost cases, the LINEX Wk-means, and the LINEX EWk-means leads to better results than Wk-means.

Table 1 The results of Wk-means, LINEX Wk-means, and LINEX EWk-means algorithms on simulated datasets.

Dataset	Algorithm	β	a	AM	Max of AM	DB	NVI
Normal	Wk-means	2.1	-	99.66	**100.0**	**0.57**	0.03
	LINEX Wk-means	1	10^{-6}	**100.0**	**100.0**	**0.57**	**0.00**
	LINEX EWk-means	-	10^{-6}	**100.0**	**100.0**	**0.57**	**0.00**
Log-Normal	Wk-means	2	-	84.60	89.00	0.89	0.74
	LINEX Wk-means	1	10^{-6}	93.91	**100.0**	0.85	0.42
	LINEX EWk-means	-	10^{-6}	**95.72**	**100.0**	**0.79**	**0.40**
Gamma	Wk-means	2	-	97.61	**98.00**	0.63	0.28
	LINEX Wk-means	1	10^{-6}	**98.00**	**98.00**	**0.62**	**0.21**
	LINEX EWk-means	-	10^{-6}	**98.00**	**98.00**	**0.62**	**0.21**
Poisson	Wk-means	1	-	94.60	**100.00**	0.43	0.21
	LINEX Wk-means	1	10^{-6}	**97.96**	**100.00**	**0.44**	**0.14**
	LINEX EWk-means	-	10^{-6}	97.18	**100.00**	**0.44**	**0.14**

Abbreviations: AM: Accuracy measure (average percent of actual clustering labels); DB: Davies–Bouldin index; NVI: Normalized variation information.

Table 2 | The results of Wk-means, LINEX Wk-means, and LINEX EWk-means algorithms on the simulated datasets with the added noisy features.

Dataset	Algorithm	β	a	AM	Max of AM	DB	NVI
Normal with three noisy feature	Wk-means	2.1	-	98.2	**100.0**	**0.62**	0.08
	LINEX Wk-means	1	10^{-6}	**100**	**100.0**	**0.60**	**0.00**
	LINEX EWk-means	-	10^{-6}	**100**	**100.0**	**0.60**	**0.00**
Log-Normal with three noisy feature	Wk-means	2	-	83.9	**100.0**	0.74	0.29
	LINEX Wk-means	1	10^{-6}	92.1	**100.0**	0.80	0.20
	LINEX EWk-means	-	10^{-6}	**94.2**	**100.0**	**0.78**	**0.17**
Gamma with three noisy feature	Wk-means	2	-	94.78	**98.00**	0.65	0.36
	LINEX Wk-means	1	10^{-6}	**98.00**	**98.00**	**0.62**	**0.21**
	LINEX EWk-means	-	10^{-6}	**98.00**	**98.00**	**0.62**	**0.21**
Poisson with a noisy feature	Wk-means	1	-	89.32	95.00	1.80	0.48
	LINEX Wk-means	1	10^{-6}	94.00	**100.0**	0.46	0.32
	LINEX EWk-means	-	10^{-6}	**96.94**	**100.0**	**0.45**	**0.18**

5.2. Performance in Real Datasets

In this part, we examine the LINEX Wk-means, the LINEX EWk-means, and the Wk-means algorithms on three real datasets and three real datasets with added noisy features. Like the previous part, each time, we run the algorithm 100 times, and we compute the averages of AM, NVI, and DB criteria in all iterations. In these datasets, the overestimating and the underestimating are important since misclassifying an entity in a special cluster might be hazardous or costly. We consider $a \in (0, 1)$ in steps of 0.1 and we choose a according to the lowest values of the criteria. We note that $a = 1$ means the highest overestimating, and $a > 1$ does not lead to good results most of the times (since it may push all of the data into one cluster). Choosing β is the same as a, but we take $\beta \in [1, 5]$ in steps of 0.1. According to the results of Tables 3 and 4, we understand that in most datasets $\beta = 1$ leads to the best results in LINEX Wk-means and LINEX EWk-means algorithms. The performances of LINEX Wk-means and LINEX EWk-means algorithms are not so different, but they seem to give better results than Wk-means algorithm. One advantage of LINEX EWk-means algorithm in comparison with LINEX Wk-means is that it has only one parameter, a. In Table 4, we add some noisy features to each dataset and repeat the clustering algorithms. The results illustrate that our algorithms are not sensitive to the noisy features like Wk-means algorithm.

Table 3 | The results of Wk-means, LINEX Wk-means, and LINEX EWk-means algorithms on three real datasets.

Dataset	Algorithm	β	a	AM	Max of AM	DB	NVI
Haberman's survival	Wk-means	1	-	81.43	95.75	3.98	0.82
	LINEX Wk-means	1	0.2	**100.0**	**100.0**	4.43	**0.00**
	LINEX EWk-means	-	0.2	75.26	76.14	**1.36**	0.96
Magic Gamma telescope	Wk-means	2	-	64.90	66.37	1.44	0.99
	LINEX Wk-means	1	0.1	**70.79**	71.25	**1.34**	**0.95**
	LINEX EWk-means	-	0.1	70.17	**71.31**	**1.34**	0.96
Seismic bumps	Wk-means	1	-	92.17	**93.42**	0.81	**1.00**
	LINEX Wk-means	1	0.1	**93.42**	**93.42**	**0.79**	**1.00**
	LINEX EWk-means	-	0.1	**93.42**	**93.42**	**0.79**	**1.00**

Table 4 | The results of algorithms in Table 3 on three real datasets with the added noisy features.

Dataset	Algorithm	β	a	AM	Max of AM	DB	NVI
Haberman's survival with three noisy feature	Wk-means	1	-	90.86	95.75	11.00	0.41
	LINEX Wk-means	1	0.2	**100.0**	**100.0**	9.78	**0.00**
	LINEX EWk-means	-	0.2	66.14	68.30	**2.34**	0.99
Magic Gamma telescope with 11 noisy feature	Wk-means	2	-	64.75	64.93	2.18	0.99
	LINEX Wk-means	1	0.1	**70.61**	**71.33**	**1.96**	**0.94**
	LINEX EWk-means	-	0.1	70.29	**71.33**	1.97	**0.94**
Seismic bumps with 19 noisy feature	Wk-means	1	-	91.08	**93.42**	0.83	**1.00**
	LINEX Wk-means	1	0.1	**93.42**	**93.42**	**0.79**	**1.00**
	LINEX EWk-means	-	0.1	**93.42**	**93.42**	**0.79**	**1.00**

6. CONCLUSIONS

In Wk-means algorithms, different weights are assigned to various features. Therefore, the less important features have lower influences in clustering results. We propose two weighted k-means algorithms with asymmetric LINEX loss function as the dissimilarity measure. They are useful when the overestimating and the underestimating are important, while they consider features with different weights. In these algorithms, the optimal centers are different from Wk-means algorithms with Euclidian dissimilarity distance. To evaluate the proposed algorithms, we investigate the results on some real and simulated datasets, and we compute some internal and external criterion to check the accuracies. The results represent good performances of the algorithms.

ACKNOWLEDGMENTS

The authors thank the editor and the reviewers of the manuscript for their excellent comments and suggestions which help to improve the quality of the content considerably.

REFERENCES

1. G.H. Ball, D.J. Hall, Behav. Sci. 12 (1967), 153–155.
2. J.Z. Huang, J. Xu, M. Ng, Y. Ye, in: Computational Methods of Feature Selection, Chapman & Hall, Boca Raton, FL, 2008, pp. 193–209.
3. R.C. de Amorim, P. Komisarczuk, in: J. Hollmén, F. Klawonn, A. Tucker, (Eds.), Advances in Intelligent Data Analysis XI. IDA. Lecture Notes in Computer Science, vol. 7619, Springer, Berlin, Heidelberg, 2012, pp. 45–55.
4. D.S. Modha, W.S. Spangler, Mach. Learn. 52 (2003), 217–237.
5. K. Kummamuru, R. Krishnapuram, R. Agrawal, in ICDM '05 Proceedings of the Fifth IEEE International Conference on Data Mining, Houston, TX, 2005, pp. 697–700.
6. A. Parsian, S.N.U.A. Kirmani, in: A. Ullah, (Ed.), Handbook of Applied Econometrics and Statistical Inference, vol. 165, CRC Press, Boca Raton, 2002, pp. 53–76.
7. N. Ahmadzadehgoli, A. Mohammadpour, M.H. Behzadi, J. Stat. Theory Appl. 17 (2018), 29–38.
8. H.R. Varian, in: L.J. Savage, S.E. Fienberg, A. Zellner, (Eds.), Studies in Bayesian Econometrics and Statistics in Honor of Leonard J. Savage, North-Holland Pub. Co., Amsterdam, 1975, pp. 195–208.
9. R.C. de Amorim, B. Mirkin, Pattern Recognit. 45 (2011), 1061–1075.
10. A. Asuncion, D.J. Newman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, 2007. http://www.ics.uci.edu/~mlearn/MLRepository.html
11. R. Reichart, A. Rappoport, in Proceedings of the Thirteenth Conference on Computational Natural Language Learning, Boulder, 2009, pp. 165–173.
12. D. Davies, D. Bouldin, IEEE Trans. Pattern Anal. Mach. Intell. 1 (1979), 224 –227.