



ELSEVIER

Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Pattern Recognition Letters 24 (2003) 2883–2893

Pattern Recognition
Letters

www.elsevier.com/locate/patrec

k^* -Means: A new generalized k -means clustering algorithm [☆]

Yiu-Ming Cheung *

Department of Computer Science, Hong Kong Baptist University, 7/F Sir Run Run Shaw Building, Kowloon Tong, Hong Kong

Received 23 July 2002; received in revised form 11 April 2003

Abstract

This paper presents a generalized version of the conventional k -means clustering algorithm [Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, 1, University of California Press, Berkeley, 1967, p. 281]. Not only is this new one applicable to ellipse-shaped data clusters without dead-unit problem, but also performs correct clustering without pre-assigning the exact cluster number. We qualitatively analyze its underlying mechanism, and show its outstanding performance through the experiments.

© 2003 Elsevier B.V. All rights reserved.

Keywords: Clustering analysis; k -Means algorithm; Cluster number; Rival penalization

1. Introduction

Clustering analysis is a fundamental but important tool in statistical data analysis. In the past, the clustering techniques have been widely applied in a variety of scientific areas such as pattern recognition, information retrieval, microbiology analysis, and so forth.

In the literature, the k -means (MacQueen, 1967) is a typical clustering algorithm, which aims to partition N inputs (also called *data points* interchangeably) $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ into k^* clusters by assigning an input \mathbf{x}_i into the j th cluster if the indicator function $I(j|\mathbf{x}_i) = 1$ holds with

$$I(j|\mathbf{x}_i) = \begin{cases} 1 & \text{if } j = \arg \min_{1 \leq r \leq k} \|\mathbf{x}_i - \mathbf{m}_r\|^2; \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Here, $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_k$ are called *seed points* or *units* that can be learned in an adaptive way as follows:

- Step 1.* Pre-assign the number k of clusters, and initialize the seed points $\{\mathbf{m}_j\}_{j=1}^k$.
- Step 2.* Given an input \mathbf{x}_i , calculate $I(j|\mathbf{x}_i)$ by Eq. (1).
- Step 3.* Only update the winning seed point \mathbf{m}_w , i.e., $I(w|\mathbf{x}_i) = 1$, by

$$\mathbf{m}_w^{\text{new}} = \mathbf{m}_w^{\text{old}} + \eta(\mathbf{x}_i - \mathbf{m}_w^{\text{old}}), \quad (2)$$

where η is a small positive learning rate.

The above Step 2 and Step 3 are repeatedly implemented for each input until all seed points converge.

[☆] This work was supported by a Faculty Research Grant of Hong Kong Baptist University with the project code: FRG/02-03/I-06.

* Tel.: +852-3411-5155; fax: +852-3411-7892.

E-mail address: ymc@comp.hkbu.edu.hk (Y.-M. Cheung).

Although the k -means has been widely applied in image processing, pattern recognition and so forth, it has three major drawbacks:

- (1) It implies that the data clusters are ball-shaped because it performs clustering based on the Euclidean distance only as shown in Eq. (1).
- (2) As pointed out in (Xu et al., 1993), there is the dead-unit problem. That is, if some units are initialized far away from the input data set in comparison with other units, they then immediately become dead without learning chance any more in the whole learning process.
- (3) It needs to pre-determine the cluster number. When k equals to k^* , the k -means algorithm can correctly find out the clustering centres as shown in Fig. 1(b). Otherwise, it will lead to an incorrect clustering result as depicted in Fig. 1(a) and (c), where some of m_j s do not locate at the centres of the corresponding clusters. Instead, they are either at some boundary points among different clusters or at points biased from some cluster centres.

In the literature, the k -means has been extended by considering the input covariance matrix in clustering via Eq. (1) so that it can work on ellipse-shaped data clusters as well as ball-shaped ones. Furthermore, there have been sev-

eral techniques proposed to solve the dead-unit problem. Frequency Sensitive Competitive Learning (FSCL) algorithm (Ahalt et al., 1990) is a typical example that circumvents the dead units by gradually reducing the winning chance of the frequent winning unit. As for the cluster number selection, some works have been done along two directions. The first one is to formulate the cluster number selection as the choice of component number in a finite mixture model. In the past, there have been some criteria proposed for model selection, such as AIC (Akaike, 1973, 1974), CAIC (Bozdogan, 1987) and SIC (Schwarz, 1978). Often, these existing criteria may overestimate or underestimate the cluster number due to the difficulty of choosing an appropriate penalty function. In recent years, a number selection criterion developed from Ying-Yang Machine has been proposed and experimentally verified in (Xu, 1996, 1997), whose computing however is laborious. The other direction invokes some heuristic approaches. For example, the typical incremental clustering gradually increases the number k of clusters under the control of a threshold value, which unfortunately is hard to be decided. Furthermore, Probabilistic Validation (PV) approach (Har-even and Brailovsky, 1995) performs clustering analysis by projecting the high-dimension inputs into one dimension via maximizing the projection indices. It has been

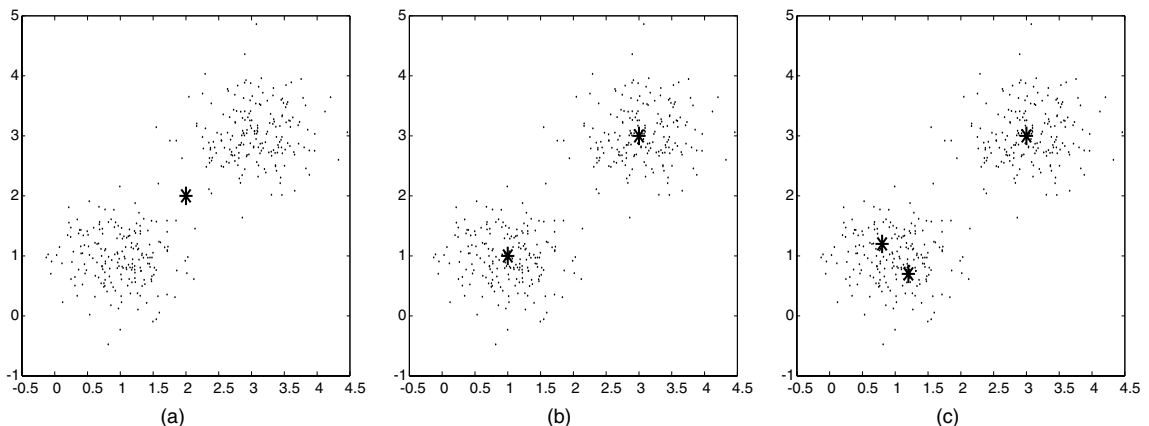


Fig. 1. The results of the k -means algorithm under two-cluster data set with (a) $k = 1$; (b) $k = 2$; (c) $k = 3$, where ‘*’ denotes the locations of the converged seed points m_j s.

shown that the PV can find out the correct number of clusters with a high probability. However, not only is this algorithm essentially suitable for the linear-separable problems only with the few number of clusters, but also requests the clusters to be well-separated with the overlap ignorable. Otherwise, its two-level clustering validation procedure becomes rather time-consuming, and the probability of finding the correct number of clusters decreases. In addition, another typical example is an improved version of FSCL named Rival Penalised Competitive Learning (RPCL) (Xu et al., 1993) that for each input, not only the winner of the seed points is updated to adapt to the input, but also its rival is de-learned by a smaller learning rate (also called *de-learning rate* hereafter). Many experiments have shown that the RPCL can select the correct cluster number by driving extra seed points far away from the input data set, but its performance is sensitive to the selection of the de-learning rate. To our best knowledge, such a rate selection so far has not been well-guided by any theoretical result.

In this paper, we will present a new clustering technique named STep-wise Automatic Rival-penalised (STAR) k -means algorithm (denoted as k^* -means hereafter), which is actually a generalization of the conventional k -means algorithm, but without its three major drawbacks as stated previously. The k^* -means consists of two separate steps. The first one is a pre-processing procedure, which assigns each cluster at least a seed point. Then, the next step is to adjust the units adaptively by a learning rule that automatically penalises the winning chance of all rival seed points in the subsequent competitions while tuning the winning one to adapt to an input. This new algorithm has a similar mechanism to RPCL in performing clustering without pre-determining the correct cluster number. The main difference is that the proposed one penalises the rivals in an implicit way, whereby circumventing the determination of the rival de-learning rate as presented in the RPCL. We have qualitatively analyzed the underlying rival-penalised mechanism of this new algorithm, and empirically shown its clustering performance on synthetic data.

2. A metric for data clustering

Suppose N inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ are independently and identically distributed from a mixture-density-of-Gaussian population:

$$p^*(\mathbf{x}; \Theta^*) = \sum_{j=1}^{k^*} \alpha_j^* G(\mathbf{x} | \mathbf{m}_j^*, \Sigma_j^*), \quad (3)$$

with

$$\sum_{j=1}^{k^*} \alpha_j^* = 1, \quad \text{and} \quad \alpha_j^* \geq 0 \quad \text{for} \quad 1 \leq j \leq k^*, \quad (4)$$

where k^* is the mixture number, $\Theta^* = \{(\alpha_j^*, \mathbf{m}_j^*, \Sigma_j^*) | 1 \leq j \leq k^*\}$ is the true parameter set, and $G(\mathbf{x} | \mathbf{m}, \Sigma)$ denotes a multivariate Gaussian density of \mathbf{x} with mean \mathbf{m} (also called *seed points* or *units*) and covariance Σ . In Eq. (3), both of k^* and Θ^* are unknown, and need to be estimated. We therefore model the inputs by

$$p(\mathbf{x}; \Theta) = \sum_{j=1}^k \alpha_j G(\mathbf{x} | \mathbf{m}_j, \Sigma_j), \quad (5)$$

with

$$\sum_{j=1}^k \alpha_j = 1, \quad \text{and} \quad \alpha_j \geq 0 \quad \text{for} \quad 1 \leq j \leq k, \quad (6)$$

where k is a candidate of mixture number, $\Theta = \{(\alpha_j, \mathbf{m}_j, \Sigma_j) | 1 \leq j \leq k\}$ is an estimator of Θ^* . We measure the distance between $p^*(\mathbf{x}; \Theta^*)$ and $p(\mathbf{x}; \Theta)$ by the following Kullback–Leibler divergence function:

$$\begin{aligned} Q(\mathbf{x}; \Theta) &= \int p^*(\mathbf{x}; \Theta^*) \ln \frac{p^*(\mathbf{x}; \Theta^*)}{p(\mathbf{x}; \Theta)} d\mathbf{x} \quad (7) \\ &= \sum_{j=1}^k \int p(j|\mathbf{x}) p^*(\mathbf{x}; \Theta^*) \ln \frac{p^*(\mathbf{x}; \Theta^*)}{p(\mathbf{x}; \Theta)} d\mathbf{x} \\ &= \sum_{j=1}^k \int p(j|\mathbf{x}) p^*(\mathbf{x}; \Theta^*) \ln \frac{p(j|\mathbf{x}) p^*(\mathbf{x}; \Theta^*)}{\alpha_j G(\mathbf{x} | \mathbf{m}_j, \Sigma_j)} d\mathbf{x} \quad (8) \end{aligned}$$

with

$$p(j|\mathbf{x}) = \frac{\alpha_j G(\mathbf{x} | \mathbf{m}_j, \Sigma_j)}{p(\mathbf{x}; \Theta)}, \quad 1 \leq j \leq k, \quad (9)$$

where $p(j|\mathbf{x})$ is the posterior probability of an input \mathbf{x} from the probability density function (pdf) j as given \mathbf{x} . It can be seen that minimizing Eq. (8) is equivalent to the maximum likelihood (ML) learning of Θ , i.e., minimizing Eq. (7), upon the fact that $\int p^*(\mathbf{x}; \Theta^*) \ln p^*(\mathbf{x}; \Theta^*) d\mathbf{x}$ is a constant irrelevant to Θ . Actually, this relation was first built in *Ying-Yang Machine* (Xu, 1995–1997), which is a unified statistical learning approach beyond ML framework in general, with a special structural design of the four Ying-Yang components. Here, we adhere to estimate Θ within the ML framework only.

It should be noted that Eqs. (3) and (5) are both the identifiable model, i.e., given a specific mixture number, $p^*(\mathbf{x}; \Theta^*) = p(\mathbf{x}; \Theta)$ if and only if $\Theta^* = \Theta$. Hence, as given $k \geq k^*$, $Q(\mathbf{x}; \Theta)$ will reach the minimum when $\bar{\Theta} = \Theta^*$, i.e., $p^*(\mathbf{x}; \Theta^*) = p(\mathbf{x}; \Theta)$, where $\bar{\Theta} = \Theta - A(\Theta)$ with $A(\Theta) = \{(\alpha_j, \mathbf{m}_j, \Sigma_j) | \alpha_j = 0, 1 \leq j \leq k\}$. Hence, Eq. (8) is an appropriate metric for data clustering by means of $p(j|\mathbf{x})$. Here, we prefer to perform clustering based on the *winner-take-all* principle. That is, we assign an input \mathbf{x} into cluster j if

$$I(j|\mathbf{x}) = \begin{cases} 1 & \text{if } j = w = \arg \max_{1 \leq r \leq k} p(r|\mathbf{x}); \\ 0 & \text{otherwise,} \end{cases} \quad (10)$$

which can be further specified as

$$I(j|\mathbf{x}) = \begin{cases} 1 & \text{if } j = w = \arg \min_r \rho_r; \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

with

$$\rho_r = \left[(\mathbf{x}_t - \mathbf{m}_r)^T \Sigma_r^{-1} (\mathbf{x}_t - \mathbf{m}_r) - \ln(|\Sigma_r^{-1}|) - 2 \ln(\alpha_r) \right]. \quad (12)$$

Consequently, minimizing Eq. (8) is approximate to minimize

$$R(\mathbf{x}; \Theta) = \sum_{j=1}^k \int I(j|\mathbf{x}) p^*(\mathbf{x}; \Theta^*) \times \ln \frac{I(j|\mathbf{x}) p^*(\mathbf{x}; \Theta^*)}{\alpha_j G(\mathbf{x} | \mathbf{m}_j, \Sigma_j)} d\mathbf{x}, \quad (13)$$

which, by the law of large number, can be further simplified as

$$R(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N; \Theta) = H - \frac{1}{N} \sum_{t=1}^N \sum_{j=1}^k I(j|\mathbf{x}_t) \times \ln[\alpha_j G(\mathbf{x} | \mathbf{m}_j, \Sigma_j)] \quad (14)$$

as N is large enough, where $H = \frac{1}{N} \times \sum_{t=1}^N \ln p^*(\mathbf{x}_t; \Theta^*)$ is a constant term irrelevant to Θ . Hence, when all inputs $\{\mathbf{x}_t\}_{t=1}^N$ are available, the learning of Θ via minimizing Eq. (14) can be implemented by the hard-cut Expectation–Maximization (EM) algorithm (Xu, 1995) in a batch way, which however needs to pre-assign the mixture number k appropriately. Otherwise, it will lead to an incorrect solution. Here, we prefer to perform clustering and parameter learning adaptively in analog with the previous k -means, but has robust clustering performance without pre-assigning the exact cluster number. The paper (Xu, 1995) has proposed an adaptive EM algorithm as well, but its convergence properties and robustness have not been well studied yet. Furthermore, the paper (Wang et al., 2003) has presented a gradient-based learning algorithm to learn the parameter set Θ via minimizing the soft version of Eq. (14), i.e., replace $I(j|\mathbf{x}_t)$ by $p(j|\mathbf{x}_t)$ in Eq. (14). Although the preliminary experiments have shown its robust performance on Gaussian-mixture clustering, it actually belongs to a batch-way algorithm, and updates *all* parameters at each time step without considering the characteristics of the metric, resulting in considerable computations needed. In Section 4, we therefore present an alternative adaptive gradient-based algorithm to minimize Eq. (14) for the parameter learning and clustering.

Before closing this section, two things should be further noted. The first one is that Eq. (14) can be degenerated to mean-square-error (MSE) function if α_r s are all forced to $1/k$, and Σ_r s are all the same. Under the circumstances, the clustering based on Eq. (11) is actually the conventional k -means algorithm. The other thing is that the term $\ln(\alpha_r)$ with $r \neq w$ in Eq. (12) is automatically decreased because of the summation constraints among α_r s in Eq. (6) when α_w is adjusted to adapt the winning of cluster w for an input \mathbf{x}_t . Consequently, all rival seed points are automatically penalised in a sense of winning chance while the winner is modified to adapt to the input \mathbf{x}_t . In the next section, we will

show that such a penalization can drive the winning chance of extra seed points in the same cluster towards zero.

3. Rival-penalised mechanism analysis of the metric

For simplicity, we consider one cluster with two seed points denoted as m_1 and m_2 , respectively. In the beginning, we assume that $\alpha_1^{(\tau)} = \alpha_2^{(\tau)}$ with $\tau = 0$, where the superscript $\tau \geq 0$ denotes the number of times that the data have been repeatedly scanned. Hence, based on the data assignment condition in Eq. (11), $m_1^{(0)}$ and $m_2^{(0)}$ divide the cluster into two regions: Regions 1 and 2 by a separating line $L^{(0)}$ as shown in Fig. 2(a). In general, the number $n_1^{(0)}$ of the inputs falling in Region 1 is different from $n_2^{(0)}$ in Region 2. Without loss of generality, we further suppose $n_1^{(0)} > n_2^{(0)}$. During data scanning, if $m_j^{(0)}$ wins to adapt to an input x_i , $\alpha_j^{(0)}$ will be increased by a unit $\Delta\alpha$ towards minimizing Eq. (14). Since $n_1^{(0)} > n_2^{(0)}$, after scanning all the data points in the cluster, the net increase of $\alpha_1^{(0)}$ will be about $(n_1^{(0)} - n_2^{(0)})\Delta\alpha$, and the net decrease of $\alpha_2^{(0)}$ will be in the same amount due to the constraint that $\alpha_1^{(0)} + \alpha_2^{(0)} = 1$. Consequently, the separating line between Region 1 and Region 2 is moved towards the right direction as shown in Fig. 2(b). That is, the area of Region 1 is being

expanded towards the right meanwhile Region 2 is being shrunk. This scenario will be always kept along with τ increase until the seed point m_2 is stabilized at the boundary of the cluster with its associated $\alpha_2 = 0$. From Eq. (11), we know that ρ_2 tends to positive infinity. That is, m_2 has actually been dead without chance to win again. Although m_2 still stays in the cluster, it cannot interfere with the learning of m_1 any more. Consequently, m_1 will gradually converge to the cluster center through minimizing Eq. (14).

In the above, we have ignored the effects of $\Sigma_j s$ in Eq. (12) for simplicity. Actually, $\Sigma_j s$ are insensitive to the gradual change of the region boundaries in comparison with $m_j s$ and $\alpha_j s$. That is, the dominant term of determining the linear moving direction is the third term in Eq. (12). Moreover, the previous analysis merely investigates a simple one-cluster case. In general, the analysis of multiple clusters is more complicated because of the interactive effects among clusters, particularly when their overlaps are considerable. Under the circumstances, the results are similar to the one-cluster case, but the extra seed points may not die at the cluster boundary. Instead, they may stay at a position with a small distance to the boundary. In Section 5, we will give out some experiments to further justify these results.

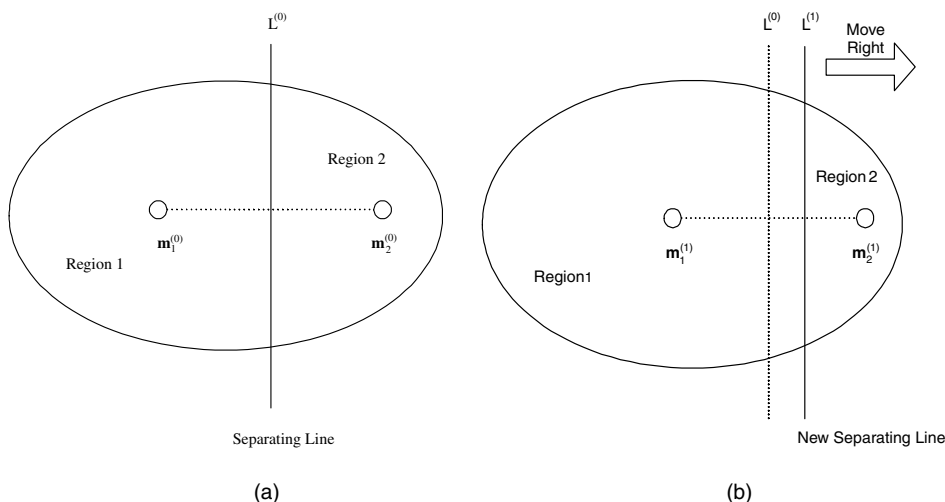


Fig. 2. The region boundaries of the seed points m_1 and m_2 that divide the cluster into two regions: Regions 1 and 2 by a separating line L with (a) the initial region boundary, and (b) the boundary after all data points in the cluster have been scanned once.

4. k^* -Means algorithm

From the results of Section 3, we know that the data assignment based on the condition in Eq. (11) can automatically penalise the extra seed points without requiring any other efforts. Hence, the k^* -means algorithm consists of two separate steps. The first step is to let each cluster acquires at least one seed point, and the other step is to adjust the parameter set Θ via minimizing Eq. (14) meanwhile clustering the data points by Eq. (11). The detailed k^* -means algorithm is given out as follows:

Step 1: We implement this step by using Frequency Sensitive Competitive Learning (Ahalt et al., 1990) because they can achieve the goal as long as the number of seed points is not less than the exact number k^* of clusters. Here, we suppose the number of clusters is $k \geq k^*$, and randomly initialize the k seed points $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_k$ in the input data set.

Step 1.1: Randomly pick up a data point \mathbf{x}_t from the input data set, and for $j = 1, 2, \dots, k$, let

$$u_j = \begin{cases} 1 & \text{if } j = w = \arg \min_r \lambda_r \|\mathbf{x}_t - \mathbf{m}_r\|; \\ 0 & \text{otherwise,} \end{cases} \quad (15)$$

where $\lambda_j = n_j / \sum_{r=1}^k n_r$, and n_r is the cumulative number of the occurrences of $u_r = 1$.

Step 1.2: Update the winning seed point \mathbf{m}_w only by

$$\mathbf{m}_w^{\text{new}} = \mathbf{m}_w^{\text{old}} + \eta(\mathbf{x}_t - \mathbf{m}_w^{\text{old}}). \quad (16)$$

Steps 1.1 and 1.2 are repeatedly implemented until the k series of u_j , $j = 1, 2, \dots, k$ remain unchanged for all \mathbf{x}_t s. Then go to Step 2. In the above, we have not included the input covariance information in Eqs. (15) and (16) because this step merely aims to allocate the seed points into some desired regions as stated before, rather than making a precise value estimate of them. Hence, we can simply ignore the covariance information to save the considerable computing cost in the estimate of a covariance matrix.

Step 2: Initialize $\alpha_j = 1/k$ for $j = 1, 2, \dots, k$, and let Σ_j be the covariance matrix of those data points with $u_j = 1$. In the following, we adaptively learn α_j s, \mathbf{m}_j s and Σ_j s towards minimizing Eq. (14).

Step 2.1: Given a data point \mathbf{x}_t , calculate $I(j|\mathbf{x}_t)$ s by Eq. (11).

Step 2.2: Update the winning seed point \mathbf{m}_w only by

$$\begin{aligned} \mathbf{m}_w^{\text{new}} &= \mathbf{m}_w^{\text{old}} - \eta \frac{\partial R}{\partial \mathbf{m}_w} \Big|_{\mathbf{m}_w^{\text{old}}} \\ &= \mathbf{m}_w^{\text{old}} + \eta \Sigma_w^{-1} (\mathbf{x}_t - \mathbf{m}_w^{\text{old}}), \end{aligned} \quad (17)$$

or simply by Eq. (16) without considering Σ_w^{-1} . In the latter, we actually update \mathbf{m}_w along the direction of $\Sigma_w^{-1} \frac{\partial R}{\partial \mathbf{m}_w}$ that forms an acute angle to the gradient-descent direction. Further, we have to update the parameters α_j s and Σ_w . The updates of the former can be obtained by minimizing Eq. (14) through a constrained optimization algorithm in view of the constraints on α_j s in Eq. (6). Alternatively, we here let

$$\alpha_j = \frac{\exp(\beta_j)}{\sum_{r=1}^k \exp(\beta_r)}, \quad 1 \leq j \leq k, \quad (18)$$

where the constraints of α_j s are automatically satisfied, but the new variables β_j s are totally free. Consequently, instead of α_j s, we can learn β_w^{new} only by

$$\begin{aligned} \beta_w^{\text{new}} &= \beta_w^{\text{old}} - \eta \frac{\partial R}{\partial \beta_w} \Big|_{\beta_w^{\text{old}}} \\ &= \beta_w^{\text{old}} + \eta(1 - \alpha_w^{\text{old}}), \end{aligned} \quad (19)$$

with the other β_j s unchanged. It turns out that α_w is exclusively increased while the other α_j s are penalised, i.e., their values are decreased. Here, please note that, although α_j s are gradually convergent, Eq. (19) always makes the updating of β increase without an upper bound upon the fact the α_w is always smaller than 1 in general. To avoid this undesirable situation, one feasible way is to subtract a positive constant c_β from all β_j s when the largest one of β_j s reaches a pre-specified positive threshold value. As for Σ_w , we update it with a small step size along the direction towards minimizing Eq. (14), i.e.,

$$\Sigma_w^{\text{new}} = (1 - \eta_s) \Sigma_w^{\text{old}} + \eta_s \mathbf{z}_t \mathbf{z}_t^T, \quad (20)$$

where $\mathbf{z}_t = \mathbf{x}_t - \mathbf{m}_w^{\text{old}}$, and η_s is a small positive learning rate. In general, the learning of a covariance matrix is more sensitive to the learning step

size than the other parameters. Hence, to make Σ_w learned smoothly, by rule of thumb, η_s can be chosen much smaller than η , e.g., $\eta_s = 0.1\eta$. Since Eqs. (11) and (17) involve Σ_j^{-1} s only rather than Σ_j s, to save computing costs and calculation stability, we therefore directly update Σ_w^{-1} by reformatting Eq. (20) in terms of Σ_w^{-1} . Consequently, we have

$$\Sigma_w^{-1\text{new}} = \frac{\Sigma_w^{-1\text{old}}}{1 - \eta_s} \left[\mathbf{I} - \frac{\eta_s \mathbf{z}_t \mathbf{z}_t^T \Sigma_w^{-1\text{old}}}{1 - \eta_s + \eta_s \mathbf{z}_t^T \Sigma_w^{-1\text{old}} \mathbf{z}_t} \right], \quad (21)$$

where \mathbf{I} is an identity matrix.

Steps 2.1 and 2.2 are repeatedly implemented until k series of $I(j|\mathbf{x}_t)$ with $j = 1, 2, \dots, k$ remain unchanged for all \mathbf{x}_t s.

5. Experimental results

We performed two experiments to demonstrate the performance of k^* -means algorithm. Experiment 1 used the 1000 data points from a mixture of three Gaussian distributions:

$$\begin{aligned} p(\mathbf{x}) = & 0.3G \left[\mathbf{x} \middle| \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0.1, & 0.05 \\ 0.05, & 0.2 \end{pmatrix} \right] \\ & + 0.4G \left[\mathbf{x} \middle| \begin{pmatrix} 1 \\ 5 \end{pmatrix}, \begin{pmatrix} 0.1, & 0 \\ 0, & 0.1 \end{pmatrix} \right] \\ & + 0.3G \left[\mathbf{x} \middle| \begin{pmatrix} 5 \\ 5 \end{pmatrix}, \begin{pmatrix} 0.1, & -0.05 \\ -0.05, & 0.1 \end{pmatrix} \right]. \end{aligned} \quad (22)$$

As shown in Fig. 3(a), the data form three well-separated clusters. We randomly initialized six seed points in the input data space, and set the learning rates $\eta = 0.001$ and $\eta_s = 0.0001$. After Step 1 of k^* -means algorithm, each cluster has been assigned at least one seed point as shown in Fig. 3(b). We then performed Step 2, resulting in α_1, α_5 and α_6 converging to 0.2958, 0.3987 and 0.3055 respectively, while the others converged to zero. That is, the seed points $\mathbf{m}_2, \mathbf{m}_3$ and \mathbf{m}_4 are the extra ones whose winning chances have been penalised to zero during the competitive learning with other seed points. Consequently, as shown in Fig. 3(c), the three clusters have been well recognized with

$$\begin{aligned} \mathbf{m}_1 = & \begin{pmatrix} 1.0087 \\ 0.9738 \end{pmatrix}, \quad \Sigma_1 = \begin{pmatrix} 0.0968, & 0.0469 \\ 0.0469, & 0.1980 \end{pmatrix} \\ \mathbf{m}_5 = & \begin{pmatrix} 0.9757 \\ 4.9761 \end{pmatrix}, \quad \Sigma_5 = \begin{pmatrix} 0.0919, & 0.0016 \\ 0.0016, & 0.0908 \end{pmatrix} \\ \mathbf{m}_6 = & \begin{pmatrix} 5.0163 \\ 5.0063 \end{pmatrix}, \quad \Sigma_6 = \begin{pmatrix} 0.1104, & -0.0576 \\ -0.0576, & 0.1105 \end{pmatrix}, \end{aligned} \quad (23)$$

while the extra seed points $\mathbf{m}_2, \mathbf{m}_3$ and \mathbf{m}_4 have been pushed to stay at the boundary of their corresponding clusters. It can be seen that this result is accordance with the analysis in Section 3.

In Experiment 2, we used 2000 data points that are also from a mixture of three Gaussians as follows:

$$\begin{aligned} p(\mathbf{x}) = & 0.3G \left[\mathbf{x} \middle| \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0.15, & 0.05 \\ 0.05, & 0.25 \end{pmatrix} \right] \\ & + 0.4G \left[\mathbf{x} \middle| \begin{pmatrix} 1 \\ 2.5 \end{pmatrix}, \begin{pmatrix} 0.15, & 0 \\ 0, & 0.15 \end{pmatrix} \right] \\ & + 0.3G \left[\mathbf{x} \middle| \begin{pmatrix} 2.5 \\ 2.5 \end{pmatrix}, \begin{pmatrix} 0.15, & -0.1 \\ -0.1, & 0.15 \end{pmatrix} \right], \end{aligned} \quad (24)$$

which results in a serious overlap among the clusters as shown in Fig. 4(a). Under the same experimental environment, we first performed Step 1, resulting in the six seed points distributed in the three clusters as shown in Fig. 4(b). Then we performed Step 2, which led to $\alpha_2 = 0.3879$, $\alpha_3 = 0.2925$, and $\alpha_6 = 0.3196$ while the others became to zero. Consequently, the corresponding converged \mathbf{m}_j s and Σ_j s were:

$$\begin{aligned} \mathbf{m}_2 = & \begin{pmatrix} 0.9491 \\ 2.4657 \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} 0.1252, & 0.0040 \\ 0.0040, & 0.1153 \end{pmatrix} \\ \mathbf{m}_3 = & \begin{pmatrix} 1.0223 \\ 0.9576 \end{pmatrix}, \quad \Sigma_3 = \begin{pmatrix} 0.1481, & 0.0494 \\ 0.0494, & 0.2189 \end{pmatrix} \\ \mathbf{m}_6 = & \begin{pmatrix} 2.5041 \\ 2.5161 \end{pmatrix}, \quad \Sigma_6 = \begin{pmatrix} 0.1759, & -0.1252 \\ -0.1252, & 0.1789 \end{pmatrix}, \end{aligned} \quad (25)$$

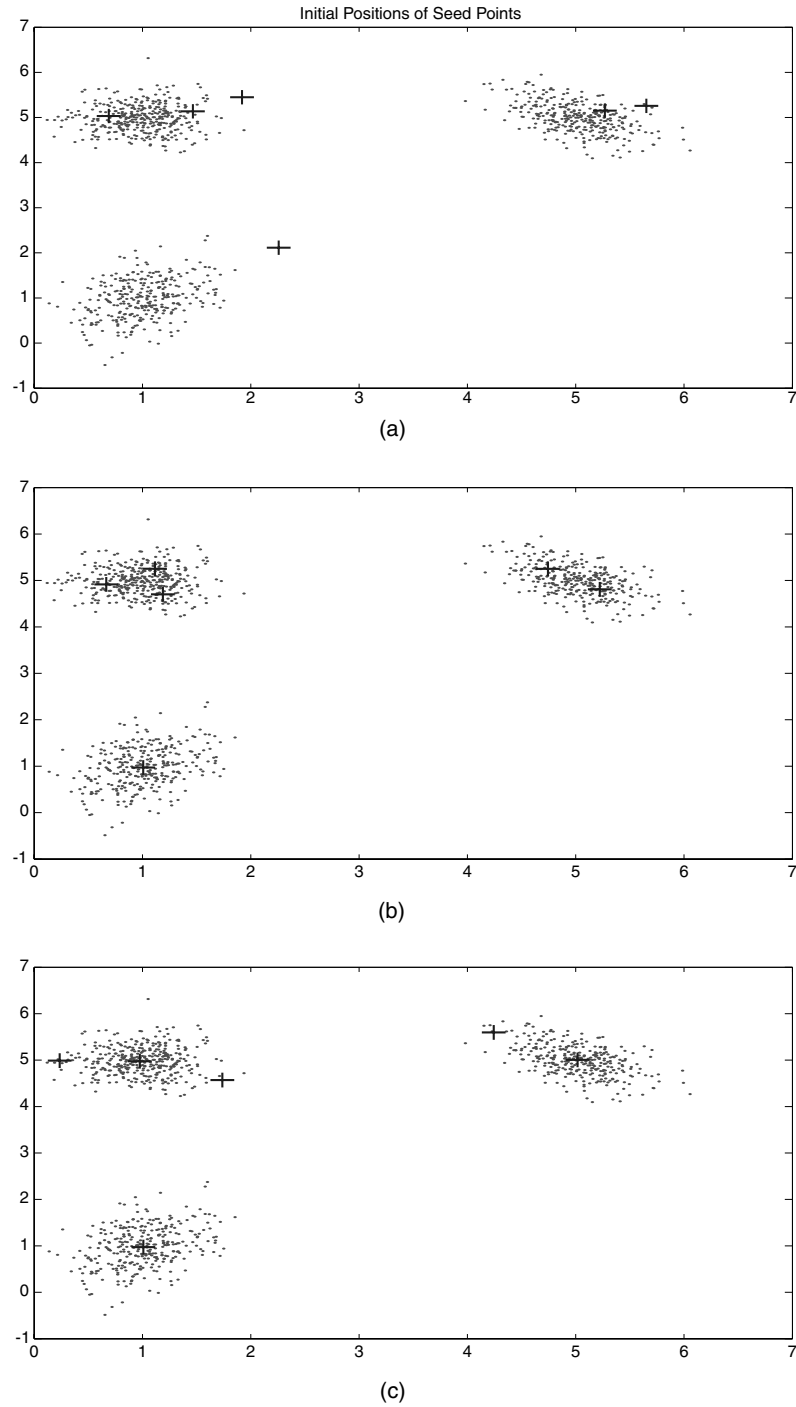


Fig. 3. The positions of six seed points marked by '+' in the input data space at different steps in Experiment 1: (a) the initial positions, (b) the positions after Step 1 of the k^* -means algorithm, and (c) the final positions after Step 2.

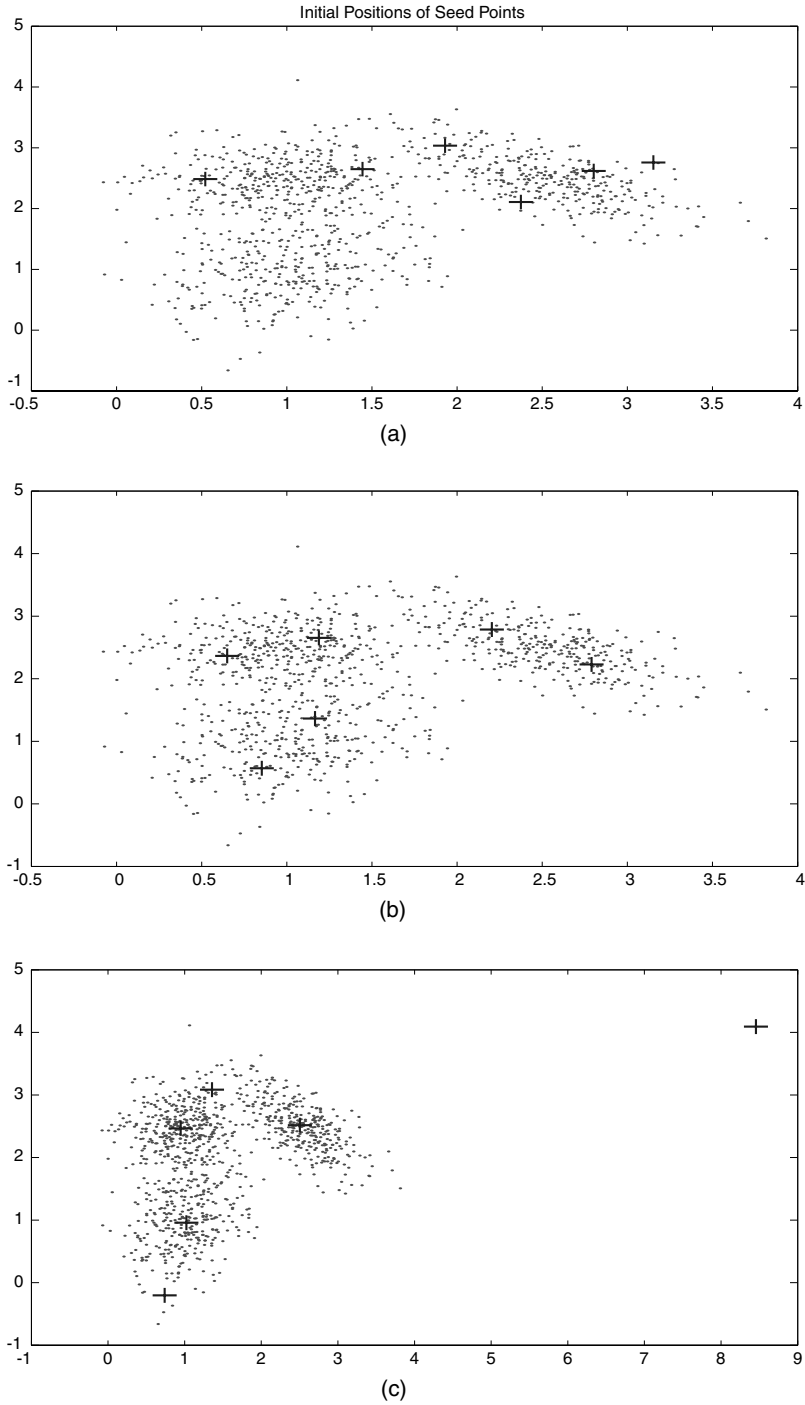


Fig. 4. The positions of six seed points marked by '+' in the input data space at different steps in Experiment 2: (a) the initial positions, (b) the positions after Step 1 of the k^+ -means algorithm, and (c) the final positions after Step 2.

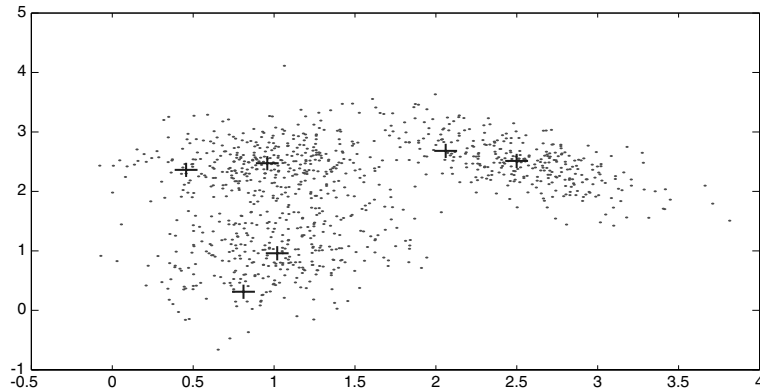


Fig. 5. The final positions of six seed points marked by '+' in the input data space, where the seed points are updated by Eq. (16).

while the other three extra seed points were stabilized at

$$\begin{aligned} \mathbf{m}_1 &= \begin{pmatrix} 0.7394 \\ -0.2033 \end{pmatrix}, & \mathbf{m}_4 &= \begin{pmatrix} 8.4553 \\ 4.0926 \end{pmatrix}, \\ \mathbf{m}_5 &= \begin{pmatrix} 2.5041 \\ 2.5166 \end{pmatrix}. \end{aligned} \quad (26)$$

As shown in Fig. 4(c), \mathbf{m}_1 and \mathbf{m}_5 have been pushed to stay at the boundary of their corresponding clusters. However, we also found that \mathbf{m}_4 had been driven far away from the input data set, but not stayed at the cluster boundary. The reason is that the main diagonal elements of Σ_4 are generally very small, i.e., those of Σ_4^{-1} become very large. Subsequently, the updating of \mathbf{m}_4 (i.e., the second term in Eq. (17)) is considerably large when the fixed learning step size η is not sufficiently small. It turns out that \mathbf{m}_4 is strongly driven to the outside far away from the correspond cluster. Actually, when we update all \mathbf{m}_i s by Eq. (16) instead of Eq. (17), all converged seed points will then finally stay within the clusters as shown in Fig. 5, where all extra seed points die near the boundaries of their corresponding clusters upon the effects of the cluster overlapping. Again, this experimental result is consistent with the analysis in Section 3.

6. Conclusion

We have presented a new generalization of conventional k -means clustering algorithm. Not

only is this new one applicable to ellipse-shaped data clusters as well as ball-shaped ones without dead-unit problem, but also performs correct clustering without pre-determining the exact cluster number. We have qualitatively analyzed its rival-penalised mechanism, and shown its outstanding clustering performance via the experiments.

References

- Ahalt, S.C., Krishnamurty, A.K., Chen, P., Melton, D.E., 1990. Competitive learning algorithms for vector quantization. *Neural Networks* 3, 277–291.
- Akaike, H., 1973. Information theory and an extension of the maximum likelihood principle. In: *Proc. Second Internat. Symposium on Information Theory*, pp. 267–281.
- Akaike, H., 1974. A new look at the statistical model identification. *IEEE Trans. Automatic Control* AC-19, 716–723.
- Bozdogan, H., 1987. Model selection and Akaike's information criterion the general theory and its analytical extensions. *Psychometrika* 52 (3), 345–370.
- Har-even, M., Brailovsky, V.L., 1995. Probabilistic validation approach for clustering. *Pattern Recognition Lett.* 16, 1189–1196.
- MacQueen, J.B., 1967. Some methods for classification and analysis of multivariate observations. In: *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1. University of California Press, Berkeley, CA, pp. 281–297.
- Schwarz, G., 1978. Estimating the dimension of a model. *Ann. Statist.* 6 (2), 461–464.
- Wang, T.J., Ma, J.W., Xu, L., 2003. A gradient BYY harmony learning rule on Gaussian mixture with automated model selection, *Neurocomputing*, in press.
- Xu, L., 1995. Ying-Yang Machine: A Bayesian–Kullback scheme for unified learning and new results on vector

- quantization. In: Proc. 1995 Internat. Conf. on Neural Information Processing (ICONIP'95), pp. 977–988.
- Xu, L., 1996. How many clusters? A Ying-Yang Machine based theory for a classical open problem in pattern recognition. In: Proc. IEEE Internat. Conf. Neural Networks, vol. 3. 1996, pp. 1546–1551.
- Xu, L., 1997. Bayesian Ying-Yang Machine, clustering and number of clusters. *Pattern Recognition Lett.* 18 (11–13), 1167–1178.
- Xu, L., Krzyżak, A., Oja, E., 1993. Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. *IEEE Trans. Neural Networks* 4, 636–648.