PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 38 (2005) 607–611

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Rapid and brief communication

Evaluation of the performance of clustering algorithms in kernel-induced feature space

Dae-Won Kim[a,*], Ki Young Lee[b], Doheon Lee[a], Kwang H. Lee[a,b]

[a]*Department of BioSystems and Advanced Information Technology Research Center, Korea Advanced Institute of Science and Technology, Guseong-dong, Yuseong-gu, 305-701, Daejeon, Republic of Korea*
[b]*Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Guseong-dong, Yuseong-gu, 305-701, Daejeon, Republic of Korea*

Received 7 September 2004; accepted 15 September 2004

Abstract

By using a kernel function, data that are not easily separable in the original space can be clustered into homogeneous groups in the implicitly transformed high-dimensional feature space. Kernel *k*-means algorithms have recently been shown to perform better than conventional *k*-means algorithms in unsupervised classification. However, few reports have examined the benefits of using a kernel function and the relative merits of the various kernel clustering algorithms with regard to the data distribution. In this study, we reformulated four representative clustering algorithms based on a kernel function and evaluated their performances for various data sets. The results indicate that each kernel clustering algorithm gives markedly better performance than its conventional counterpart for almost all data sets. Of the kernel clustering algorithms studied in the present work, the kernel average linkage algorithm gives the most accurate clustering results.

Keywords: Clustering; Kernel; *k*-means; Fuzzy *c*-means; Average linkage; Mountain algorithm

1. Introduction

A clustering has emerged as a popular technique for pattern recognition, image processing, and data mining. The kernel-based classification in the feature space not only preserves the inherent structure of groups in the input space, but also simplifies the associated structure of the data [1]. Since Girolami first developed the kernel *k*-means clustering algorithm for unsupervised classification [2], several studies have demonstrated the superiority of kernel clustering algorithms over other approaches to clustering [3,4].

Users of kernel clustering methods are often left wondering to what extent kernel clustering algorithms are superior to conventional algorithms with regard to the data distribution and which clustering algorithm is the most improved by reformulation in the kernel-induced feature space. In this paper, we evaluate the performance of kernel clustering algorithms with a view to providing answers to these questions. To our knowledge, this is the first such comparison of kernel clustering algorithms for general purpose clustering. We consider four well-known clustering algorithms: the *k*-means, fuzzy *c*-means, average linkage, and mountain algorithms. We compare the performances of these four algorithms with those of the kernel *k*-means algorithm, the kernel fuzzy *c*-means algorithm, and formulations of the average linkage and mountain algorithms based on a kernel function. These comparisons are made over a variety of data sets.

* Corresponding author. Tel.: +82 42 869 4353; fax: +82 42 869 8680.
E-mail address: dwkim@bisl.kaist.ac.kr (D.-W. Kim).

0031-3203/$30.00
doi:10.1016/j.patcog.2004.09.006

2. Kernel clustering algorithms

Given an unlabeled data set $X = \{x_1, \ldots, x_n\}$ in the d-dimensional space R^d, let $\Phi : R^d \to H$ be a non-linear mapping function from this input space to a high-dimensional feature space H. By applying the non-linear mapping function Φ, the dot product $x_i \cdot x_j$ in the input space is mapped to $\Phi(x_i) \cdot \Phi(x_j)$ in the feature space. The key notion in kernel-based learning is that the mapping function Φ need not be explicitly specified. The dot product $\Phi(x_i) \cdot \Phi(x_j)$ in the high-dimensional feature space can be calculated through the kernel function $K(x_i, x_j)$ in the input space R^d.

2.1. Kernel k-means algorithm

Given an unlabeled data set X and a mapping $\Phi : R^d \to H$, the k-means algorithm in the high-dimensional feature space iteratively searches for k clusters by minimizing the function J [2,3]:

$$J(X) = \sum_{i=1}^{k} \sum_{j=1}^{n} \|\Phi(x_j) - v_i^{\Phi}\|^2, \tag{1}$$

where the ith cluster centroid $v_i^{\Phi} = n_i^{-1} \sum_{j=1}^{n} z_{ij} \Phi(x_j)$ and $n_i = \sum_{j=1}^{n} z_{ij}$. Here z_{ij} indicates whether data point x_j belongs to the ith cluster; specifically, $z_{ij} = 1$ if it belongs to the ith cluster and 0 otherwise. The key notion in the kernel k-means algorithm lies in the calculation of the distance in the feature space. The distance between $\Phi(x_j)$ and v_i^{Φ} in the feature space is calculated through the kernel in the input space:

$$\begin{aligned}
&\|\Phi(x_j) - v_i^{\Phi}\|^2 \\
&\quad = \Phi(x_j) \cdot \Phi(x_j) - 2\Phi(x_j) \cdot \frac{1}{n_i} \sum_{k=1}^{n} z_{ik} \Phi(x_k) \\
&\qquad + \frac{1}{n_i} \sum_{k=1}^{n} z_{ik} \Phi(x_k) \cdot \frac{1}{n_i} \sum_{l=1}^{n} z_{il} \Phi(x_l) \\
&\quad = \Phi(x_j) \cdot \Phi(x_j) - \frac{2}{n_i} \sum_{k=1}^{n} z_{ik} \Phi(x_k) \Phi(x_j) \\
&\qquad + \frac{1}{n_i^2} \sum_{k=1}^{n} \sum_{l=1}^{n} z_{ik} z_{il} \Phi(x_k) \Phi(x_l) \\
&\quad = K(x_j, x_j) - \frac{2}{n_i} \sum_{k=1}^{n} z_{ik} K(x_k, x_j) \\
&\qquad + \frac{1}{n_i^2} \sum_{k=1}^{n} \sum_{l=1}^{n} z_{ik} z_{il} K(x_k, x_l).
\end{aligned} \tag{2}$$

Therefore, the objective function can be rewritten as Eq. (3), and the process of updating clusters are repeated until there is no significant improvement in J between consecutive iterations.

$$J(X) = \sum_{i=1}^{k} \sum_{j=1}^{n} \left(K(x_j, x_j) - \frac{2}{n_i} \sum_{k=1}^{n} z_{ik} K(x_k, x_j) + \frac{1}{n_i^2} \sum_{k=1}^{n} \sum_{l=1}^{n} z_{ik} z_{il} K(x_k, x_l) \right). \tag{3}$$

The kernel k-means algorithm lacks the step in which cluster centroids are updated so as to reassign the data point to the closest cluster because the reassignment can be made without calculating the centroids due to the implicit mapping via the kernel function in Eq. (2).

2.2. Kernel fuzzy c-means algorithm

The kernel fuzzy c-means algorithm in the feature space by a mapping Φ minimizes the function J_m [4]:

$$J_m(X) = \sum_{i=1}^{c} \sum_{j=1}^{n} (\mu_{ij})^m \|\Phi(x_j) - v_i^{\Phi}\|^2, \tag{4}$$

where μ_{ij} is the membership degree of data point x_j to the ith fuzzy cluster, and m is a fuzziness coefficient. The ith cluster centroid is $v_i^{\Phi} = \sum_{j=1}^{n} (\mu_{ij})^m \Phi(x_j) / \sum_{j=1}^{n} (\mu_{ij})^m$. The k-means algorithm repeatedly updates the k clusters at each successive iteration, whereas the fuzzy c-means algorithm iteratively updates the new membership degree μ_{ij} at each iteration. The update of μ_{ij} in the feature space is defined through the kernel in the input space as follows:

$$\mu_{ij} = \left(\sum_{k=1}^{c} \left(\frac{\|\Phi(x_j) - v_i^{\Phi}\|^2}{\|\Phi(x_j) - v_k^{\Phi}\|^2} \right)^{1/(m-1)} \right)^{-1}, \tag{5}$$

where

$$\begin{aligned}
&\|\Phi(x_j) - v_i^{\Phi}\|^2 \\
&\quad = \Phi(x_j) \cdot \Phi(x_j) - 2\Phi(x_j) \cdot \frac{\sum_{k=1}^{n} (\mu_{ik})^m \Phi(x_k)}{\sum_{k=1}^{n} (\mu_{ik})^m} \\
&\qquad + \frac{\sum_{k=1}^{n} (\mu_{ik})^m \Phi(x_k)}{\sum_{k=1}^{n} (\mu_{ik})^m} \cdot \frac{\sum_{l=1}^{n} (\mu_{il})^m \Phi(x_l)}{\sum_{l=1}^{n} (\mu_{il})^m} \\
&\quad = \Phi(x_j) \cdot \Phi(x_j) - \frac{2 \sum_{k=1}^{n} (\mu_{ik})^m \Phi(x_k) \Phi(x_j)}{\sum_{k=1}^{n} (\mu_{ik})^m} \\
&\qquad + \frac{\sum_{k=1}^{n} \sum_{l=1}^{n} (\mu_{ik})^m (\mu_{il})^m \Phi(x_k) \Phi(x_l)}{(\sum_{k=1}^{n} (\mu_{ik})^m)^2} \\
&\quad = K(x_j, x_j) - \frac{2 \sum_{k=1}^{n} (\mu_{ik})^m K(x_k, x_j)}{\sum_{k=1}^{n} (\mu_{ik})^m} \\
&\qquad + \frac{\sum_{k=1}^{n} \sum_{l=1}^{n} (\mu_{ik})^m (\mu_{il})^m K(x_k, x_l)}{(\sum_{k=1}^{n} (\mu_{ik})^m)^2}.
\end{aligned} \tag{6}$$

Similar to the kernel k-means algorithm, the kernel fuzzy c-means algorithm does not need to calculate the cluster centroids because the centroid information is considered in updating the membership degree μ_{ij}.

2.3. Kernel average linkage algorithm

Compared to *k*-means-type algorithms, the average linkage algorithm is more flexible with regard to the cluster shape but has much greater time and space complexities. Based on the average distance between two clusters $F_n = \{x_1, \ldots, x_n\}$ and $F_m = \{y_1, \ldots, y_m\}$, the algorithm iteratively merges two closest clusters until a single cluster is obtained. The kernelization of the average linkage algorithm is simpler and more intuitive than those of the *k*-means-type algorithms. Given a mapping Φ, the distance between two clusters $F_n^{\Phi} = \{\Phi(x_1), \ldots, \Phi(x_n)\}$ and $F_m^{\Phi} = \{\Phi(y_1), \ldots, \Phi(y_m)\}$ in the feature space is calculated as

$$\delta_{Avg}(F_n^{\Phi}, F_m^{\Phi}) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \|\Phi(x_i) - \Phi(y_j)\|^2, \tag{7}$$

where

$$\begin{aligned} \|\Phi(x_i) - \Phi(x_j)\|^2 &= \Phi(x_i) \cdot \Phi(x_i) - 2\Phi(x_i)\Phi(x_j) \\ &\quad + \Phi(x_j)\Phi(x_j) \\ &= K(x_i, x_i) - 2K(x_i, x_j) \\ &\quad + K(x_j, x_j). \end{aligned} \tag{8}$$

The iterative merging procedure in the feature space continues until all data points have been merged into a single cluster or the number of merged groups reaches at prespecified number of clusters *k*.

2.4. Kernel mountain algorithm

The mountain algorithm estimates the cluster centroids by constructing and destroying the mountain function on a grid space. The mountain function indicates the potential that each grid point has to be a cluster centroid. To reduce the computational complexities of the original algorithm, we employed the subtractive mountain algorithm in which the mountain function is calculated on data points rather than grid points.

Given a mapping Φ, the mountain function at a data point $\Phi(x_i)$ in the feature space is defined as

$$M(\Phi(x_i)) = \sum_{j=1}^{n} \mathrm{e}^{-\alpha\|\Phi(x_i) - \Phi(x_j)\|^2}, \tag{9}$$

where $\|\Phi(x_i) - \Phi(x_j)\|^2 = K(x_i, x_i) - 2K(x_i, x_j) + K(x_j, x_j)$. A higher value of $M(\Phi(x_i))$ indicates that $\Phi(x_i)$ has more data points $\Phi(x_j)$ near to it in the feature space. After calculating the mountain values, the data point whose mountain value is $M_1^* = Max_i[M(\Phi(x_i))]$ is selected as the first cluster centroid. Subsequent centroids are found using the following modified mountain function:

$$\begin{aligned} &\widehat{M}^j(\Phi(x_i)) \\ &\quad = \widehat{M}^{j-1}(\Phi(x_i)) - M_{j-1}^* \sum_{j=1}^{n} \mathrm{e}^{-\beta\|\Phi(x_i) - \Phi(x_{j-1}^*)\|^2} \quad (10) \\ &\quad = \widehat{M}^{j-1}(\Phi(x_i)) - M_{j-1}^* \\ &\qquad \times \sum_{j=1}^{n} \mathrm{e}^{-\beta(K(x_i,x_i) - 2K(x_i,x_{j-1}^*) + K(x_{j-1}^*, x_{j-1}^*))}, \quad (11) \end{aligned}$$

where $\widehat{M}^j$ is the new mountain function, $\widehat{M}^{j-1}$ is the old mountain function, M_{j-1}^* is the maximum value of $\widehat{M}^{j-1}$, and x_{j-1}^* is the newly found centroid.

3. Experimental results

To test the various kernel clustering algorithms, we applied the four conventional clustering algorithms as well as their kernel versions to 10 widely used data sets and compared the performances of the algorithms. The data sets employed were the BENSAID (49 data/3 clusters) [5], DUNN (90 data/2 clusters) [5], IRIS (150 data/3 clusters) [5], ECOLI (336 data/7 clusters), CIRCLE, BLE-3, BLE-2, UE-4, UE-3, and ULE-4 data sets. This selection of data sets includes various types of clusters, such as hyperspherical and hyperellipsoidal, and balanced and unbalanced types (Fig. 1). The parameters used in the *k*-means and fuzzy *c*-means algorithms were a termination criterion of $\varepsilon = 0.001$ and a weighting exponent of $m = 2.0$ [5]. The initial centroids were uniformly distributed across the data set [5]. The parameters used for the mountain algorithm were $\alpha = 5.4$ and $\beta = 1.5$. The RBF kernel function was used in all four kernel clustering algorithms due to its superiority over other kernel functions [2].

The clustering accuracy achieved by each clustering algorithm for each of the 10 data sets is listed in Table 1. On the whole, the conventional *k*-means and fuzzy *c*-means algorithms showed similar performances for each data set, with average accuracies of 73.60% and 74.39%, respectively. In comparison to these two algorithms, the average linkage algorithm showed better clustering performance (average 83.33%); in particular, it achieved 100.0% accuracy for four of the ten data sets. In agreement with previous works, the present results, particularly those for the unbalanced and ellipsoidal data sets (e.g., BENSAID or BLE-2), show that the average linkage algorithm can handle a greater range of cluster shapes than the *k*-mean-type algorithms. However, it showed substantially different behavior when applied to similar data sets; for example it achieved accuracies of 56.0% and 100.0% for the BLE-3 and BLE-2, respectively, and accuracies of 71.45% and 100.0% for the UE-4 and UE-3 data sets. Such discrepancies arise because the average linkage algorithm is sensitive to the order in which data are presented. Compared to the other algorithms, the mountain

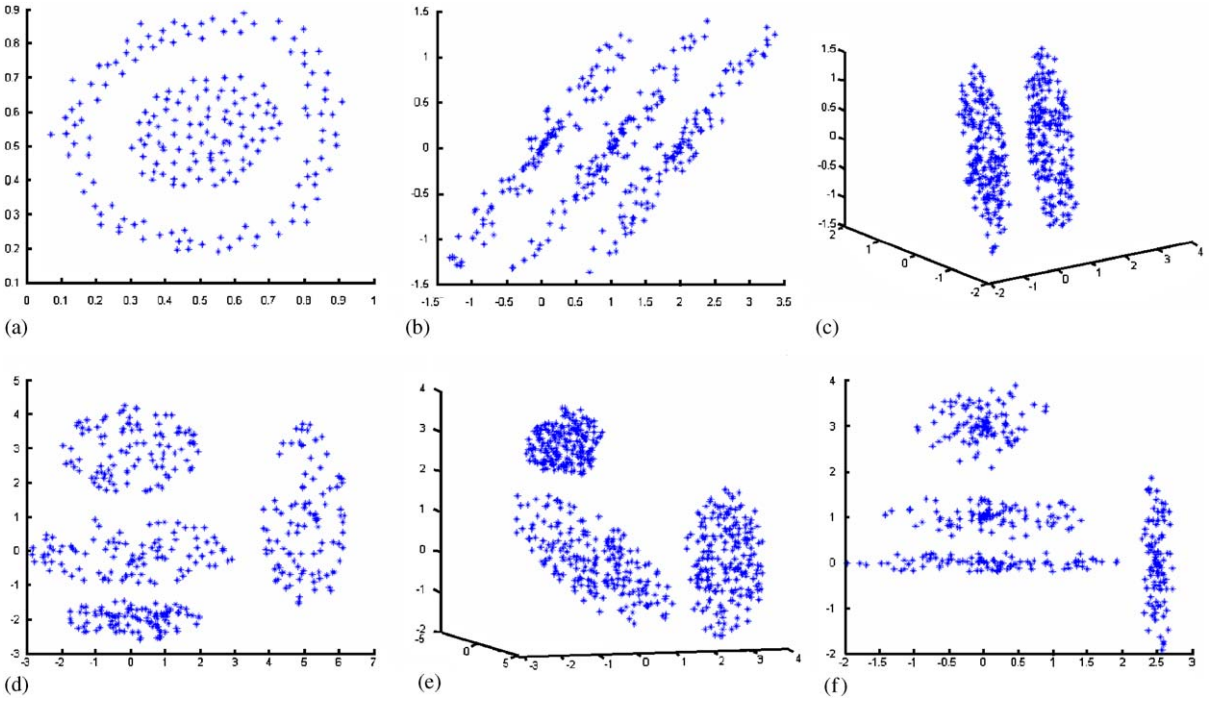

Fig. 1. Two- and three-dimensional data sets used in our evaluation: (a) CIRCLE, (b) BLE-3, (c) BLE-2, (d) UE-4, (e) UE-3, (f) ULE-4.

Table 1
Clustering accuracy (%) achieved by each clustering algorithm for 10 data sets

Data set	Conventional				Kernel			
	k-means	FCM	Average	Mountain	*k*-means	FCM	Average	Mountain
BENSAID	79.59	73.47	100.0	85.71	83.67	93.88	100.0	100.0
DUNN	70.00	70.00	100.0	83.33	71.11	95.56	100.0	100.0
IRIS	89.33	89.33	90.67	52.67	96.00	93.33	89.33	93.33
ECOLI	42.86	49.11	76.49	51.19	68.75	61.01	77.38	69.05
CIRCLE	50.76	52.79	62.44	55.84	100.0	93.40	82.74	62.94
BLE-3	65.67	65.67	56.00	70.33	76.33	74.67	100.0	71.67
BLE-2	88.50	87.75	100.0	85.25	100.0	94.00	100.0	100.0
UE-4	77.25	66.00	71.45	73.50	100.0	98.50	100.0	84.75
UE-3	95.83	95.00	100.0	51.17	98.83	96.67	100.0	95.67
ULE-4	76.25	94.75	76.25	96.25	98.00	96.25	100.0	96.25
Avg. (%)	73.60	74.39	83.33	70.52	89.27	89.73	94.95	87.37

algorithm exhibited more unstable performance. The average accuracy of the conventional mountain algorithm was 70.52%.

The clustering results obtained using the kernel clustering algorithms are listed on the right side of Table 1. It is evident that the kernel clustering algorithms give markedly better performance than the conventional algorithms. On average, the kernel *k*-means and kernel fuzzy *c*-means algorithms were about 15% more accurate than their conventional counterparts, and the kernel average linkage and kernel mountain algorithms were approximately 12% and 17% more accurate than the conventional algorithms, respectively. The clustering performance of the kernel *k*-means and kernel fuzzy *c*-means algorithm provided significantly better clustering performance for data sets that have previously been shown low performance, such as CIRCLE, BLE-2, UE-4, and ULE-4 [5]. In addition, for the CIRCLE and IRIS data sets, which contain ring-shaped clusters and overlapping

clusters, respectively, these algorithms gave better clustering results than the kernel average linkage algorithm. The kernel average linkage algorithm successfully classified seven of the 10 data sets, giving accuracies of 100.0%. Of particular note are the results for the BLE-3, UE-4, and ULE-4 data sets, for which the conventional average linkage algorithm gave accuracies of 50–70% but the kernel version classified with 100% accuracy. In terms of the total accuracy, the kernel average linkage algorithm was the most accurate clustering algorithm (94.95%). The kernel mountain algorithm gave the greatest enhancement of clustering performance for the mountain algorithm (17% improvement), with accuracies of more than 90% for six data sets. Notably, the ranking of the conventional clustering algorithms in terms of overall accuracy was preserved in their kernel versions: the kernel average linkage algorithm was the most accurate and the kernel mountain algorithm the least accurate.

4. Conclusions

Compared to the corresponding conventional clustering algorithms, the kernel clustering algorithms showed better clustering results for almost all data sets. The kernel *k*-means algorithm was significantly more accurate than its conventional counterpart, particularly when applied to data sets that have been shown low performance to date. The kernel fuzzy *c*-means algorithm achieved $>90\%$ accuracy for eight of the 10 data sets. Overall, the kernel average linkage algorithm gave the most accurate clustering results. The kernelization of the mountain algorithm improved the clustering accuracy to a degree similar to that achieved for the other algorithms. The kernel average linkage algorithm is the most appropriate to use when no prior knowledge on the characteristics of the data set is available.

Acknowledgements

This work was supported by the Korean Systems Biology Research Grant (M1-0309-02-0002) from the Ministry of Science and Technology. We would like to thank Chung Moon Soul Center for BioInformation and BioElectronics and the IBM SUR program for providing research and computing facilities.

References

[1] K.-R. Muller, et al., An introduction to kernel-based learning algorithms, IEEE Trans. Neural Networks 12 (2) (2001) 181–202.
[2] M. Girolami, Mercer kernel-based clustering in feature space, IEEE Trans. Neural Networks 13 (3) (2002) 780–784.
[3] R. Zhang, A.I. Rudnicky, A large scale clustering scheme for kernel *k*-means, in: The 16th International Conference on Pattern Recognition, 2002, pp. 289–292.
[4] Z.-d. Wu, W.-x. Xie, Fuzzy *c*-means clustering algorithm based on kernel method, in: The Fifth International Conference on Computational Intelligence and Multimedia Applications, 2003, pp. 1–6.
[5] J.C. Bezdek, et al., Fuzzy Models and Algorithms for Pattern Recognition and Image Processing, Kluwer Academy Publishers, Boston, 1999.