

Is It Rational To Partition A Data Set Using Kernel-Clustering?

Kaushik Sarkar

Department of Electronics & Communication Engineering
Narula Institute of Technology (JIS Group)
81, Nilgunj Road, Agarpara, Kolkata-700109, India
(Email: kaushik.srkr@gmail.com)

Nikhil R. Pal

Electronics and Communication Sciences Unit
Indian Statistical Institute
203, B.T. Road, Calcutta-700108
(Email: nikhil@isical.ac.in)

Abstract—Many interesting papers have been written in the recent past on kernel clustering and many attractive results have also been demonstrated. Here we question the rationality behind such clustering approaches. Using simple data sets we argue and demonstrate that it is not a good idea to find clusters in the kernel space when the objective is to look for clusters in the original data because in the kernel space the data may have a different geometry from that in the original feature space. In particular we demonstrate the following : (1) improper choice of the number of clusters may lead to very counterintuitive clusters (e.g., instead of merging nearby clusters, it may merge clusters that are far from each other) and (2) improper choice of kernel parameters has a significant effect on the extracted clusters and it can even impose arbitrary cluster structures that are undoubtedly absent in the original data. However, we definitely do not imply that kernel clustering can never produce desirable results. In fact, kernel clustering could be useful provided we can choose right kernel parameters. But the process being unsupervised, we do not have a solution to this issue yet. In this study, for illustration, we use one variant of the kernel Fuzzy C-Means (KFCM) clustering algorithm in conjunction with Polynomial kernels.

Keywords: Fuzzy C-means, FCM, Kernel fuzzy C-means, feature space, Kernel space.

I. INTRODUCTION

A cluster seeking algorithm partitions a given data set $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^p$ into c number of clusters so that members of the same cluster are similar to one another and members of different cluster are dissimilar. Clustering has many applications covering almost all areas of science and technology [1], [2] and hence there are many algorithms [1], [3], [4], [5], [6] and the research efforts are still going on to develop more effective algorithms. Some algorithms use probabilistic concepts, while others use fuzzy and/or possibilistic modeling [3], [4], [7].

Crisp clustering methods assume that each data point belongs to one and only one cluster, which may not be true in some real life applications such as land-cover classification. In such cases, fuzzy clustering is considered a better alternative for extracting sub-structures in the data. In a fuzzy cluster, a data point may partially belong to more than one cluster and the membership degrees/grades lie in $[0,1]$. The most popular fuzzy clustering algorithm is the Fuzzy C-Means (FCM) [3] which works fine when the data set has hyperspherical clusters. However, if a pair of clusters is not linearly separable (eg.,

two nested spherical-shells), such algorithms usually fail. To identify such clusters there have been many extensions/variants of the FCM algorithm [8], [9], [10], [11]. In the recent past there have been many attempts to kernelize clustering algorithms. These algorithms are demonstrated to be useful in clustering data such as a disk surrounded by a shell, which are otherwise difficult to cluster [5], [12], [13], [14], [15], [16], [17]. These algorithms project the data implicitly into a high dimensional space with a hope that the "clusters" in the original space (we shall call it feature space) will be well separated in the kernel space. These algorithms then find the clusters in the kernel space.

There are different types of kernelized version of FCM. We shall consider a version which assumes that the cluster centers/prototypes are in the Kernel space [18], [19], [20], [21]. This family of algorithms requires an inverse mapping of prototypes from the kernel space to feature space. This type of algorithms will be called as KFCM-K (Kernel FCM with prototypes in the kernel space).

The objective of any clustering algorithm is to find the clusters (i.e., natural groups) in the *original* data space. Does a kernel clustering algorithm do so? Sometimes 2-Dimensional data sets are used to demonstrate the success of a kernel clustering algorithm. Usually some "suitable" kernel parameter values are chosen and results are demonstrated. Do we know if the algorithm will work for a wide range of choices of parameters? We shall see that even for some simple data sets, the answer to the previous question is "No". When the data sets are in 2-D we can visualize easily the performance. But when the data are in high dimension visual assessment is difficult. For such cases, usually the cluster centroids or the partition are used in conjunction with a labeled data set (the data set is for a classification problem) to assess the resubstitution or classification error. This approach makes an assumption that number of natural clusters is equal to the number of classes. But this may not necessarily be true as clustering and classification are philosophically two different problems. For example, a two class problem may have four distinct clusters and we may not know it (A high dimensional generalization of the Ex-OR problem). If the resubstitution or classification error is low, we should be happy, but if not, we cannot make any inference because the number of clusters may be equal

to the number of classes but the clustering could be poor or the number of clusters could be more than the number of classes and that may lead to poor classification performance. So assuming number of classes equal to the number of clusters and then checking the resubstitution error or classification error is not a good way of justifying kernel clustering.

Here we investigate whether clustering in the kernel space is justified or not. Our investigation suggests that Kernel clustering can lead to strange (counterintuitive) results if the choice of number of clusters and/or the choice of kernel parameters are not appropriate. We discuss and demonstrate this issue using some simple data sets.

II. KERNELIZED VERSION FCM

A data set may have clusters in it, but it may be difficult for an algorithm like FCM to extract those clusters. One reasons for this may be that a pair of clusters is not linearly separable. In such a case, the original data can be mapped to a higher dimensional space (kernel space) with a hope that clusters will be easily identifiable [19], [20], [21]. For this a non linear mapping function f is used such that $\phi : R^d \rightarrow H$, where H is the kernel space. Typically kernel-clustering algorithms use the kernel trick that takes the advantage of the fact that the dot products in the kernel space can be represented by a Mercer kernel K where $K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$, where, $\mathbf{x}, \mathbf{y} \in R^p$. Thus the distance/similarity in the kernel space can be replaced by the Mercer kernel. Some of the popular kernel functions are:

- 1) RBF kernel : $e^{-\|\mathbf{x}-\mathbf{y}\|^2/\sigma^2}, \sigma^2 > 0$
- 2) Polynomial kernel : $(\mathbf{x}^T \cdot \mathbf{y} + a)^b, a \geq 0, b \in N$
- 3) Hyper-tangent : $\tanh(\mathbf{x}^T \cdot \mathbf{y} + a), a \geq 0$

In this study, we shall consider only one Kernel based FCM algorithm, KFCM-K and use only the Polynomial kernel (PK). We shall not consider the other versions, such as KFCM-F [18], [5], [22], where cluster centers are considered in the feature space itself.

A. KFCM-K [13], [14], [15], [16], [18]

Here the prototypes are located in the kernel space, so the pre-images of prototypes in the feature space have to be computed by taking the inverse mapping from kernel space to feature space. KFCM-K algorithm minimizes the following objective function at (1).

$$Q = \sum_{i=1}^c \sum_{k=1}^n u_{ik}^m \|\phi(\mathbf{x}_k) - \mathbf{v}_i\|^2 \quad (1)$$

subject to the constraints

- 1) $u_{ik} \in [0, 1] \forall i, k$
- 2) $0 < \sum_{k=1}^n u_{ik} < n \forall i$
- 3) $\sum_{i=1}^c u_{ik} = 1 \forall k$

In (1) $\mathbf{v}_i$ is i^{th} prototype in the kernel space. Using the Lagrange Multiplier technique, optimization of Q with respect to $\mathbf{v}_i$, results in

$$\mathbf{v}_i = \frac{\sum_{k=1}^n u_{ik}^m \phi(\mathbf{x}_k)}{\sum_{k=1}^n u_{ik}^m} \quad (2)$$

Similarly, the expression for the membership values are derived as

$$u_{ik} = \frac{1}{\sum_{j=1}^c \left(\frac{\|\phi(\mathbf{x}_k) - \mathbf{v}_i\|}{\|\phi(\mathbf{x}_k) - \mathbf{v}_j\|} \right)^{\frac{2}{m-1}}}; i = 1, \dots, c; k = 1, \dots, n \quad (3)$$

where

$$\begin{aligned} \|\phi(\mathbf{x}_k) - \mathbf{v}_i\|^2 &= (\phi(\mathbf{x}_k) - \mathbf{v}_i)^T (\phi(\mathbf{x}_k) - \mathbf{v}_i) \\ &= \phi(\mathbf{x}_k)^T \phi(\mathbf{x}_k) - 2\phi(\mathbf{x}_k)^T \mathbf{v}_i + \mathbf{v}_i^T \mathbf{v}_i. \end{aligned} \quad (4)$$

Using the expression for the prototypes (2) into the expression (4), one gets

$$\begin{aligned} \|\phi(\mathbf{x}_k) - \mathbf{v}_i\|^2 &= K(\mathbf{x}_k, \mathbf{x}_k) - 2 \frac{\sum_{j=1}^n u_{ij}^m K(\mathbf{x}_k, \mathbf{x}_j)}{\sum_{j=1}^n u_{ij}^m} \\ &\quad + \frac{\sum_{j=1}^n \sum_{l=1}^n u_{ij}^m u_{il}^m K(\mathbf{x}_j, \mathbf{x}_l)}{(\sum_{j=1}^n u_{ij}^m)^2} \end{aligned} \quad (5)$$

Following, [18], [14], the expression for the prototypes in the feature space $V = \{\tilde{\mathbf{v}}_i; i = 1, 2, \dots, c\}$ with Polynomial kernel is given in (6).

$$\tilde{\mathbf{v}}_i = \frac{\sum_{k=1}^n u_{ik}^m (\mathbf{x}_k^T \tilde{\mathbf{v}}_i + a)^{b-1} \mathbf{x}_k}{(\tilde{\mathbf{v}}_i^T \tilde{\mathbf{v}}_i + a)^{b-1} \sum_{j=1}^n u_{ij}^m} \quad (6)$$

Algorithm KFCM-K

- 1) Fix $c, t_{max}, m > 1$ and $\epsilon > 0$, the tolerance level for termination.
- 2) Initialize membership matrix U .
- 3) Repeat

Update $U = [u_{ik}]$ according to equation (3) via (5).

until $\sum_{i=1}^c \sum_{k=1}^n |u_{ik}^{(t+1)} - u_{ik}^{(t)}| < \epsilon$ is satisfied. (7)

- 4) Return U .

- 5) Initialize prototypes V

- 6) Repeat

Update $\tilde{\mathbf{v}}_i$ s according to equation (6) for Polynomial kernel

until $\sum_{i=1}^c \sum_{k=1}^n |\tilde{v}_{k,i}^{(t+1)} - \tilde{v}_{k,i}^{(t)}| < \epsilon$ is satisfied. (8)

- 7) Return V .

Note that, there are many other equivalent termination criteria that can be used in place of (7) and (8).

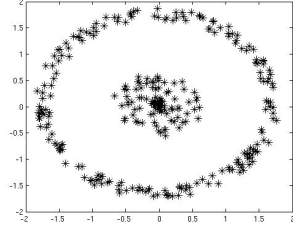
III. EXPERIMENTAL STUDIES

We use three synthetic data sets, all in R^2 : Ring, Five-Cluster, and Uniform-Disc.

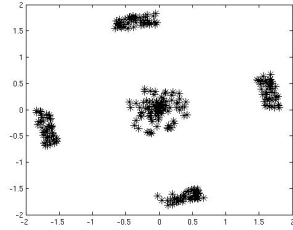
- **Ring** : The Ring Data set is depicted in Fig.1a, where the inner disk contains 100 points and the outer ring contains 200 points.

- **Five-Cluster** : Fig.1b shows the Five-Cluster data set where the inner disk, like Ring contains 100 data points forming a cluster. Surrounding this cluster, there are four smaller clusters, each with 50 points.
- **Uniformdisk** : Fig.1c depicts the Uniform data set with 441 points that are uniformly distributed. This data set does not have any cluster in it.

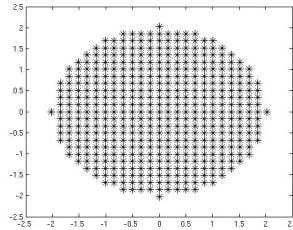
For each data set, X , every feature x is normalized by $\frac{(x-\mu)}{\sigma}$, where μ and σ are the mean and standard deviation of x as estimated from the given data set. This protocol is followed in [18].



(a) Ring dataset



(b) FiveCluster dataset



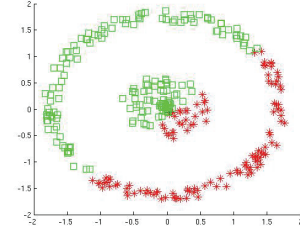
(c) Uniform dataset

Fig. 1: The three Synthetic datasets used in this study

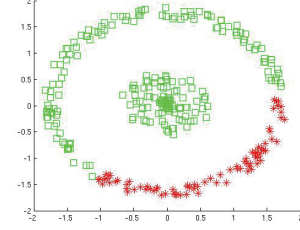
A. Kernel Clustering with Polynomial Kernel

We study the behavior of FCM, and KFCM-K with Polynomial kernel on the three data sets Ring, Five-Cluster, and Uniform. We use $m = 1.2, \epsilon = 10^{-8}, t_{max} = 100$. For the Polynomial Kernel, we have experimented with different choices for $a \in \{1, 2, 3, 4, 5, 7, 10, 15\}$ and $b \in \{2, 4, 6, 8, 10, 12, 14, 16\}$, a total of 64 combinations. The initial prototypes are taken randomly from the input data set.

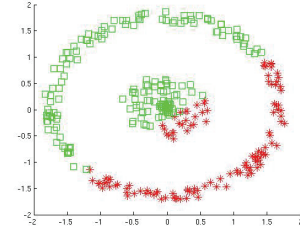
First we consider the Ring data, Fig.1a. Clearly there is a disk-shaped cluster at the center. Many authors share the



(a) FCM



(b) KFCM-K (a=1, b=2)



(c) KFCM-K (a=15, b=2)

Fig. 2: Clustering of Ring data with FCM and KFCM-K for $c=2$

view that the shell around the disk is also a cluster. We also share the same view here. For this data set with $c=2$, FCM, as expected, divides the data set as shown in Fig.2a. For some choices of a and b , the KFCM-K behaves similar to FCM, but for none of the 64 choices of parameters, KFCM-K could extract the shell as a separate cluster. Sometimes, part of the shell is extracted as a cluster and the remaining data points as another cluster. One such partition is exhibited in Fig. 2b. We also noticed that when a is high and b is low, then for this data set KFCM-K behaves like FCM. Fig 2c shows one such partition with $a=15$ and $b=2$. For this data set we find that the choice of a and b has a significant influence on the clustering output. Moreover, in this case, KFCM-K failed to produce the desired partition. Hence we do not make any further analysis on this data set.

Next we focus our attention on the Five-Cluster data set (Fig.1b), which has five distinct clusters. As expected, FCM finds all the five clusters correctly when $c=5$ (Fig.3g), so does KFCM-K with $a=1, b=2$ (Fig.3h). Fig.3, also compares the results with $c=2, 3$, and 4. For example, when $c=4$ (with $a=1, b=2$), the FCM result is quite intuitive (Fig.3e), while KFCM-K merges two clusters, which are on the *opposite sides* of the central disk, into one clusters (Fig. 3f). Thus KFCM-K

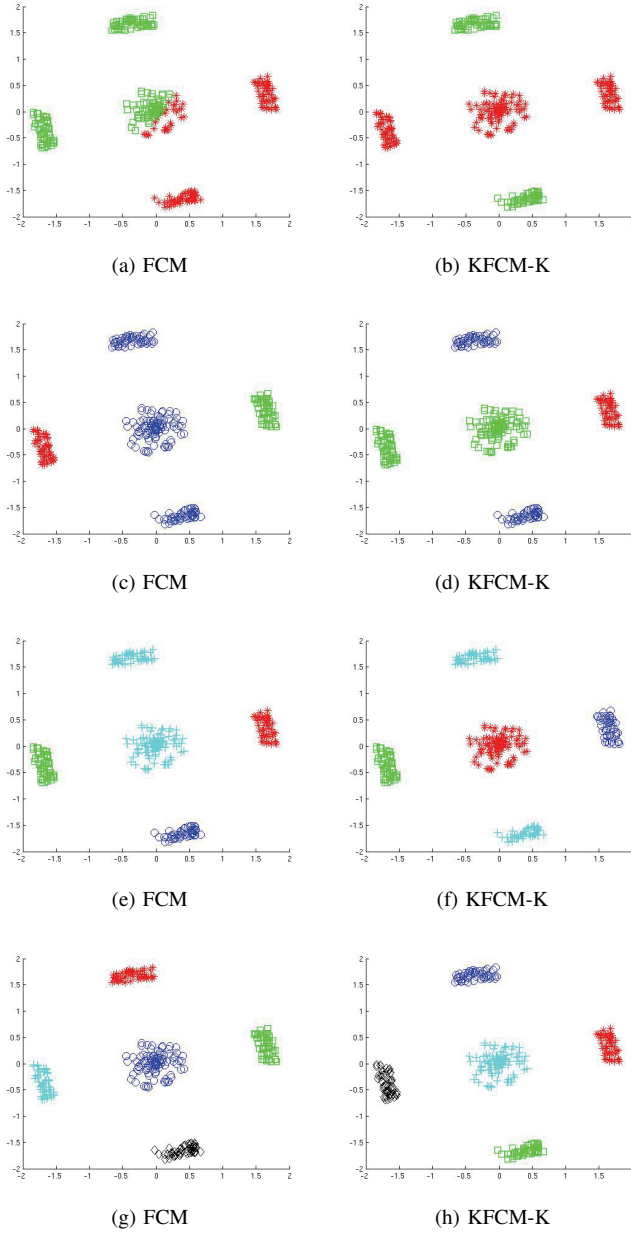


Fig. 3:
(a) FCM Clusters with $c=2$; (b)KFCM-K Clusters with $c=2$;
(c) FCM Clusters with $c=3$; (d)KFCM-K Clusters with $c=3$;
(e) FCM Clusters with $c=4$; (f)KFCM-K Clusters with $c=4$;
(g) FCM Clusters with $c=5$; (h)KFCM-K Clusters with $c=5$;

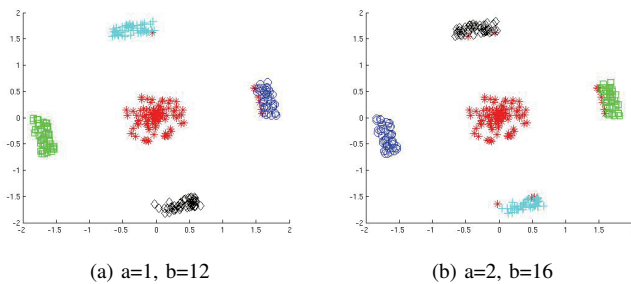


Fig. 4: Few bad clusters on the Five-Cluster dataset using KFCM-K

TABLE I: Value of Correlations for different kernel parameters [a,b=Kernel parameters, Q=Quality, PC=Pearson Correlation, KC=Kendall Correlation, SP=Spearman Correlation]

DATA	a,b	Q	a,b	Q	a,b	Q
FiveCluster	1,2	Good	5,2	Good	10,2	Good
	1,4	Good	5,4	Good	10,4	Good
	1,6	Good	5,6	Good	10,6	Good
	1,8	Good	5,8	Good	10,8	Good
	1,10	Good	5,10	Good	10,10	Good
	1,12	Bad	5,12	Good	10,12	Good
Uniform Disk	1,2	Good	5,2	Good	10,2	Good
	1,4	Medium	5,4	Good	10,4	Good
	1,6	Bad	5,6	Good	10,6	Good
	1,8	Bad	5,8	Medium	10,8	Good
	1,10	Bad	5,10	Bad	10,10	Good
	1,12	Bad	5,12	Bad	10,12	Medium

merges two small clusters when they are quite far from each other. Note that, we have run the clustering algorithms with different initializations, and we got similar results in several runs. However, for $c=2$ and $c=3$, FCM produces intuitively acceptable results (Fig.3a and Fig. 3c), while KFCM-K does not (Fig.3b and Fig. 3d) as it merges two clusters which are not nearby. In this case, since the data are in 2-D we can assess the results visually. But for high dimensional data, we may not know the desired number of clusters and hence we may end up with very undesirable results (merging of two distant clusters) and we would not even know that. This is one facet of the problems associated with kernel clustering.

The other problems is associated with the choice of kernel parameters. To demonstrate this issue, we consider $c=5$, the best choice for number of clusters for this data set. Then we ran KFCM-K for all 64 combinations of a and b as mentioned above. Table I, summarizes the results. Table I reveals that when $a=1$, and b is high (12 or higher), KFCM-K fails to extract the correct partition. Fig.4 (a) reveals that with $a=1$ and $b=12$, a few points of the cyan (plus) and blue (circle) clusters get merged with the central disk. Similarly, with $a=2$ and $b=14$ or higher, the partition starts departing from the desirable one (Fig.4 (b)). Thus we find that the choice of kernel parameters is crucial to the success of the algorithm even for very simple data sets. Since, for high dimensional data we would not know the sub-structures present in the original data, we shall not know whether KFCM-K has found a good solution or not. One can make the same arguments with FCM, but the differences are : (a) with improper choices of c , the number of clusters, FCM splits or merges clusters in such a manner that they agree with our intuition and FCM does not impose strange geometric structures on the data and (b) FCM does not have these two extra parameters, which may change the geometry of the data.

Now we consider the data set, Uniform (Fig.1c). Clearly Uniform does not have any sub-structure or cluster. Hence any clustering algorithm, with $c=2$, should partition the data into approximately two halves, where the approximate line partitioning the data may have one of many possible orientations. Fig. 5a shows that FCM does the same. Similarly, KFCM-K also does the same with $a=1$, $b=2$ (Fig.5b); as b increases, one of the clusters starts becoming bigger (Fig.5c). With $a=1$,

$b=12$ or above, KFCM-K extracts a *nice shell* (Fig. 5d) as one of the clusters. This behavior is also repeated with other choices of a . In fact, when a is increased, KFCM-K starts extracting a shell for further high values of b . This is a very serious limitation of the KFCM-K algorithm. As an illustration in Fig. 6 we show the membership values of different points to the cluster they belong to. Figure 6 corresponds to Fig. 5d. In Fig. 6, for each point we find the maximum membership value and depending on its cluster label we plot it either in green (square) or in red (circle). Figure 6 shows that for most of the points in the central big cluster the membership values to the cluster are very high, while for most points belonging to the imposed outer-shell cluster, the membership values are above 0.6. There are some points in the bigger cluster with memberships varying from values slightly higher than 0.5 to further high values suggesting a kind of gradual transition from one cluster to the other. In this data set, there is no reason to assume such a strange cluster structure.

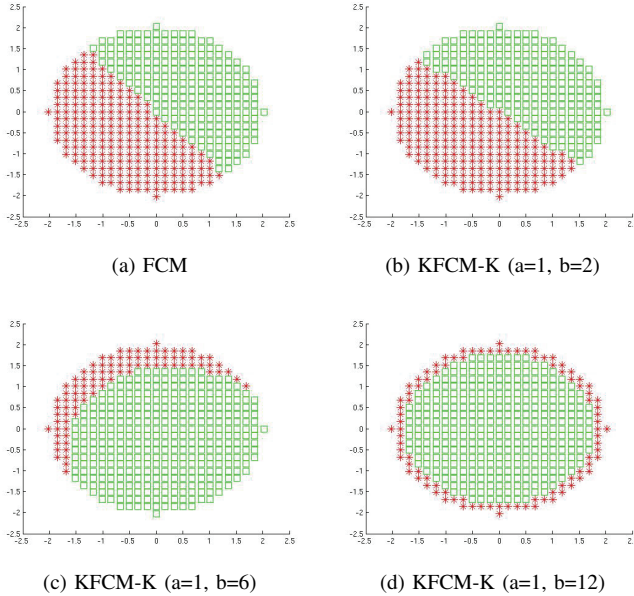


Fig. 5: Results on the Uniform dataset by FCM and KFCM-K for different parameters

Table I summarizes the visual assessment of quality of the partitions generated by kernel clustering for some illustrative sets of a and b . In Table I, Medium indicates minor departure from ideal partition while Bad indicates significant departure from the ideal partition.

So what do we learn from these experiments?

- Knowledge of the number of clusters present in the data is essential for the success of KFCM-K; otherwise, some cluster may include data points which are far from each other. In particular when c is less than the actual number clusters, KFCM-K may merge two subclusters which are far from each other in the actual data (feature) space ignoring nearby clusters. Thus producing counterintuitive clusters.

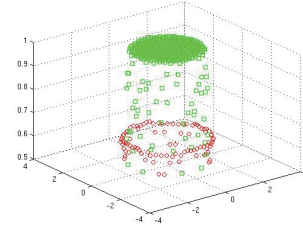


Fig. 6: Pictorial representation of the maximum membership values to the clusters.

- When a data set does not have any cluster structure, depending on the choice of kernel-parameters, it can impose very artificial substructures (e.g., the KFCM-K extracts a shell from the uniform data). To summarize, KFCM-K may impose arbitrary structures on the data.
- Even when the number of clusters is known, appropriate choice of the kernel parameters is essential to get the desirable results.

These problems are quite serious because for any real life application, data are usually in a high dimensional space. But why do all these happen? We think, this happens because we are trying to extract clusters in a space which may alter the geometry of the original data. This does not mean that for some data set KFCM-K type clustering cannot help. For example, if we have two touching clusters and in the projected space the two clusters become separated, then it would be easy to find the clusters in the kernel space. There could be other interesting cases where kernelized version could be useful. For example, using RBF kernel with an appropriate choice of the kernel width, it is possible to extract the shell and the central disk as two different clusters [18], but we need to use the right kernel with the right choice of parameters.

IV. DISCUSSION AND CONCLUSION

The performance of the kernel clustering algorithm, KFCM-K raises a very fundamental question : Should we really cluster a given data set in the kernel space? Although in the kernel space the data points might be more sparsely distributed, we cannot say that the clusters that we find in the kernel space will represent the same groups as those in the original feature space. We have illustrated this using a few synthetic data sets. In some of the studies involving kernel clustering the results are justified by matching the clusters with the classes [18]. In other words, one starts with a data set on k classes where every point has an associated class label indicating the class that the data point comes from. Such a data set is then clustered into $c=k$ clusters ignoring the class labels. Finally, one assesses how good the partition matches with class structure. If there is a good match we should be happy. But we need to remember that cluster structure in a data set may be quite different from the structure imposed by class labels. The situation is actually more complicated because most kernel needs specification of some parameters. We have demonstrated that the choice of the kernel parameters has a

significant impact on the performance of clustering algorithm. When the problem is of classification, we can use different ways, such as cross-validation, to choose a desirable value for the kernel parameters. But for cluster seeking problems, it is difficult to choose the optimal parameters because the problem is unsupervised.

We have demonstrated / justified that Kernel clustering may lead to very unexpected results. When the goal of clustering a data set is to find the clusters / substructures in the original data (feature) space, unless one is careful, kernelization may impose undesirable structures on the data and hence the clusters obtained in the kernel space may not exhibit the structure of the original data. But, this does not mean that kernel clustering cannot be useful at all. If one wants to cluster in the kernel space, then proper choice of the parameters of the kernel function has to be used. We do not have a mechanism to choose the desirable parameters and currently we are investigating this issue. We hope to report our findings on this in near future.

REFERENCES

- [1] M.R. Anderberg, Cluster Analysis for Application, Academic Press, New York, 1973.
- [2] E. Backer, A.K. Jain, A clustering performance measure based on fuzzy set decomposition, IEEE Trans. Pattern Anal. Mach. Intell. 3 (1) (1981) 6674.
- [3] J. C. Bezdek, Pattern Recognition With Fuzzy Objective Function Algorithms. New York: Plenum, 1981.
- [4] Nikhil R. Pal, Kuhu Pal, James M. Keller and James C. Bezdek, A Possibilistic Fuzzy c-Means Clustering Algorithm, IEEE Transactions on Fuzzy Systems, Vol. 13, No. 4, August 2005.
- [5] D.Q. Zhang, S.C. Chen, Clustering incomplete data using Kernel-based Fuzzy C-Means algorithm, Neural Processing Letters 18 (3) (2003) 155162.
- [6] I.S. Dhillon, Y. Guan, B. Kulis, A unified view of kernel k-means, spectral clustering and graph partitioning, Technical Report TR-04-25, UTCS, 2005.
- [7] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Academic Press, 2003.
- [8] J. C. Bezdek, R. J. Hathaway and N. R. Pal, Norm-induced shell-prototypes (NISF) clustering, *Neural, Parallel and Scientific Computations*, Vol. 3, pp 431-450, 1995.
- [9] E. E. Gustafson and W. Kessel, Fuzzy Clustering with a Fuzzy Covariance Matrix, Proc. 1978 IEEE CDC, IEEE Press, 1979. Piscataway, NJ, 761-766.
- [10] Dav, R. N. and Bhaswan, K. (1992). Adaptive fuzzy c-shells clustering and detection of ellipses, IEEE Trans. on Neural Networks, 3(5), 643-662.
- [11] Krisnapuram, R., Frigui, H. and Nasroui, O. (1995). Fuzzy and possibilistic shell clustering algorithms and their application
- [12] Z. D. Wu, W. X. Xie, J. P. Yu, Fuzzy c-means clustering algorithm based on kernel method, in: Proc. Of the Internat. Conf. on Computational Intelligence and Multimedia Applications, 2003, pp. 49-54.
- [13] D. Zhang, S. Chen, Fuzzy clustering using kernel method, in: Proc. of the Internat. Conf. on Control and Automation, 2002, pp. 123127.
- [14] S. Zhou, J. Gan, Mercer kernel fuzzy c-means algorithm and prototypes of clusters, in: Proc. of Conf. on Internat. Data Engineering and Automated Learning, Vol. 3177, 2004, pp. 613618.
- [15] L. Zeyu, T. Shiwei, X. Jing, J. Jun, Modified FCM clustering based on kernel mapping, in: Proc. of the Internat. Society for Optical Engineering, Vol. 4554, 2001, pp. 241245.
- [16] J. H. Chiang, P. Y. Hao, A new kernel-based fuzzy clustering approach: support vector clustering with cell growing, IEEE Transactions on Fuzzy Systems 11 (4) (2003) 518-527.
- [17] D. W. Kim, K. Y. Lee, D. Lee, K. H. Lee, Evolution of the performance of clustering algorithms kernel-induced feature space, Pattern Recognition 38 (4) (2005) 607-611.
- [18] Daniel Graves, Witold Pedrycz, Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study, Fuzzy Sets and Systems, 161 (2010) 522-543.
- [19] R. Herbrich, Learning Kernel Classifiers, MIT Press, Cambridge, MA, 2002.
- [20] K.R. Muller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf, An introduction to kernel-based learning algorithms, IEEE Transactions on Neural Networks 12 (2) (2001) 181201.
- [21] B. Scholkopf, A. Smola, K.R. Muller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (1998) 12991319.
- [22] H. Shen, J. Yang, S. Wang, X. Liu, Attribute weighted Mercer kernel based fuzzy clustering algorithm for general non-spherical datasets, Soft Computing 10 (11) (2006) 10611073.
- [23] I.S. Dhillon, Y. Guan and B. Kulis, Kernel k-means: spectral clustering and normalized cuts, KDD 04: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, NY, USA (2004), pp. 551556.
- [24] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. Pattern Recognition, 41(1):176-190, January 2008.
- [25] F.R. Bach, M.I. Jordan, Learning spectral clustering, Technical Report UCB/CSD-03-1249, EECS Department, University of California, Berkeley, 2003.
- [26] F.R. Bach, M.I. Jordan, Learning Spectral Clustering, With Application To Speech Separation, The Journal of Machine Learning Research, 7, p.1963-2001, 12/1/2006
- [27] N. Cristianini, J.S. Taylor, A. Elisseeff and J.S. Kandola, On kernel-target alignment, NIPS (2001), pp. 367373.
- [28] I. Fischer, I. Poland, New methods for spectral clustering, Technical Report IDSIA-12-04, IDSIA, 2004.
- [29] M. Meila, J. Shi, Learning segmentation by random walks, in: NIPS, 2000, pp. 873879.
- [30] B. Kulis, S. Basu, I.S. Dhillon and R. Mooney, Semi-supervised graph clustering: a kernel approach, ICML 05: Proceedings of the 22nd International Conference on Machine Learning, ACM Press, New York, NY, USA (2005), pp. 457464.
- [31] N. Lanckriet, N. Cristianini, P. Bartlett, L. El-Ghoui, and M.I Jordan. Learning the kernel matrix using semi-definite programming. In International Conference on Machine Learning (ICML 2002), 2002.
- [32] X. Yin, S. Chen, E. Hu, D. Zhang. Semi-supervised clustering with metric learning: An adaptive kernel method. Pattern Recognition, Volume 43, Issue 4, April 2010, Pages 1320-1333
- [33] M. Filippone. Dealing with non-metric dissimilarities in fuzzy central clustering algorithms. International Journal of Approximate Reasoning, 50(2):363-384, February 2009.
- [34] John W. Sammon Jr, A nonlinear Mapping for Data Structure Analysis, IEEE Transactions on Computers, Vol. C-18, No. 5, pp 401-409, May 1969.