AN UNSUPERVISED KERNEL BASED FUZZY C-MEANS CLUSTERING ALGORITHM WITH KERNEL NORMALISATION

Zhou, Shang-Ming

http://hdl.handle.net/10026.1/20380

10.1142/s1469026804001379
International Journal of Computational Intelligence and Applications
World Scientific Pub Co Pte Lt

AN UNSUPERVISED KERNEL BASED FUZZY C-MEANS CLUSTERING ALGORITHM WITH KERNEL NORMALIZATION

SHANG-MING ZHOU
Department of Computer Science, University of Essex,
Colchester CO4 3SQ, United Kingdom
szhoup@essex.ac.uk

JOHN Q. GAN
Department of Computer Science, University of Essex,
Colchester CO4 3SQ, United Kingdom
jqgan@essex.ac.uk

In this paper, a novel procedure for normalizing Mercer kernel is suggested firstly. Then, the normalized Mercer kernel techniques are applied to the fuzzy c-means (FCM) algorithm, which leads to a normalized kernel based FCM (NKFCM) clustering algorithm. In the NKFCM algorithm, implicit assumptions about the shapes of clusters in the FCM algorithm is removed so that the new algorithm possesses strong adaptability to cluster structures within data samples. Moreover, a new method for calculating the prototypes of clusters in input space is also proposed, which is essential for data clustering applications. Experimental results on several benchmark datasets have demonstrated the promising performance of the NKFCM algorithm in different scenarios.

Keywords: kernel normalization; clustering algorithm; fuzzy c-means; Mercer kernel; unsupervised data clustering.

1. Introduction

Since Vapnik introduced support vector machines (SVM) [1], the Mercer kernel based learning has received much attention in the machine learning community [2][3][4][5][6][7]. The protruding characteristics of a kernel based method is that nonlinear information processing can be carried out by means of linear techniques in an implicit high-dimensional feature space mapped by a nonlinear transformation from original input space. However, in contrast to the wide investigation of kernel based methods, very few efforts have made to introduce the kernel treatment into fuzzy clustering [8][9][10].

The objective of this paper is to normalize the traditional kernel function and apply the normalized kernel to the FCM algorithm [11], which will improve the performance of the FCM algorithm and remove some of its constraints. The FCM algorithm has been applied in various areas, however, it makes implicit assumptions concerning the shape and size of the clusters, i.e., the clusters are hyper-spherical and of approximately the same size. Some efforts have been made to avoid these assumptions. The Mahalanobis distance is used in the Gustafson-Kessel (GK) fuzzy clustering algorithm [11][12] instead of the Euclidean distance, in which the clusters are in fact assumed to be hyperelliptical. In order to detect non-hyperspherical structural subsets, Bezdek et al. [13] defined linear cluster structures of different dimensions in the fuzzy clustering process. In [14], Jerome et al. proposed a probabilistic relaxation fuzzy clustering algorithm based on the estimation of probability density function without assumptions about the size and shape of clusters. Hoeppner developed a special fuzzy shell-clustering algorithm to detect rectangular contours [15]. This paper first proposes a novel method to normalize the traditional kernel function and obtain a normalized kernel whose value possesses clear geometric interpretation, i.e. the correlation coefficient of two feature vectors in feature space. Furthermore, a normalized kernel based (NKFCM) clustering algorithm is developed to identify naturally occurring clusters while preserving the associated information about the relations between the clusters, which would remove the implicit assumption of hyper-spherical or ellipsoidal clusters within input data samples. By using the kernel normalization, the NKFCM algorithm would become more stable and some operations can be simplified, some bizarrerie within the original patterns could be relieved, and finally better performance will be achieved. Different from regression and classification applications, one of the main problems in using kernel based method for unsupervised clustering is that it is usually difficult to obtain the prototypes of clusters in both feature space and input space [16]. A new method for calculating cluster prototypes in input space is developed in the proposed NKFCM algorithm.

The organization of this paper is as follows. The next section describes the Mercer kernel normalization procedure. Section 3 addresses the NKFCM clustering algorithm, including the objective function in terms of kernels, the formulation for calculating the optimal input space partition, the expression of cluster prototypes in feature space, and the formulation for calculating cluster prototypes in input space. The proposed algorithm is experimentally evaluated in section 4. Section 5 provides conclusions with discussions.

2. Mercer kernel normalization

Generally speaking, in a Mercer kernel based method, in order to increase the separability of patterns, the observed data points in input space $\Re^P$ are often mapped into a high dimensional feature space Γ by using a nonlinear function $\Phi(\cdot)$, which is usually difficult to be expressed. Fortunately, this problem can be avoided by constructing kernel functions directly. There is a very useful trick for computing inner products in feature space using kernel functions $k(\cdot,\cdot)$ [1] [17] based on Mercer theorem [18], which states that any positive integral operator as a kernel $k(\cdot,\cdot)$ can be expanded in its eigenfunctions ψ_j, i.e.,

$$k(x,y)=\sum_{j=1}^{N}\lambda_j\psi_j(x)\psi_j(y) \quad (\lambda_j>0) \tag{1}$$

thus the mapping Φ can be expressed by

$$\Phi(x)=\left(\sqrt{\lambda_1}\psi_j(x),\sqrt{\lambda_N}\psi_N(x),\cdots\right)^T \tag{2}$$

i.e., $k(x,y)=\Phi(x)^T\cdot\Phi(y)$. The Mercer theorem makes it possible to develop a kernel based method without using an explicit expression of mapping function Φ. Moreover, if a mapping function $\Psi(\cdot)$ is defined as

$$\Psi(x)=\frac{\Phi(x)}{\|\Phi(x)\|} \tag{3}$$

then a normalized kernel is obtained as follows,

$$\begin{aligned} n(x,y) &:= \Psi(x)^T\cdot\Psi(y) \\ &= \frac{k(x,y)}{\sqrt{k(x,x)k(y,y)}} \end{aligned} \tag{4}$$

It can be seen that, $n(x,y)$ is the correlation coefficient of feature vectors $\Phi(x)$ and $\Phi(y)$ in feature space. Interestingly, $n(x,y)$ possesses the following properties:

1) $-1\le n(x,y)\le 1$;
2) $n(x,y)=\cos(\alpha)$, where α is the angle between the vectors $\Phi(x)$ and $\Phi(y)$;
3) $n(x,x)=1$, no matter what the expression of kernel function $k(x,x)$ is;
4) if $k(x,x)=1\,\forall x$, then $n(x,y)=k(x,y)$;
5) The distance between $\Psi(x)$ and $\Psi(y)$ can be evaluated as,

$$\begin{aligned} D(x,y)^2 &= \|\Psi(x)-\Psi(y)\|^2 \\ &= \Psi(x)^T\cdot\Psi(x)-2\Psi(x)^T\cdot\Psi(y)+\Psi(y)^T\cdot\Psi(y) \\ &= 2(1-n(x,y)) \end{aligned} \tag{5}$$

no matter what the expression of kernel function $k(x,y)$ is.

The above properties will make some expressions in a kernel based method simpler and more efficient. Furthermore, they would improve the performance of the kernel based method. In this paper, Gaussian kernels and polynomial kernels are considered separately, which are defined individually by:

$$k(x,y)=\exp\left(\frac{-\|x-y\|^2}{\sigma^2}\right) \tag{6}$$

and

$$k(x,y)=\left((x\cdot y)+\theta\right)^d \tag{7}$$

where σ is the width of the Gaussian kernel, θ and d are the offset and exponential parameters of the polynomial kernel respectively. Tax and Duin pointed out that the polynomial kernel in the form of

$k(x,y) = ((x \cdot y) + 1)^d$ cannot result in good tight description of the clusters by adjusting parameter *d* [19]. It is found that the offset parameter θ in the polynomial kernel has different influence on the outcome of clustering from parameter *d*. Hence, we adopt the general form (7) of polynomial kernel in this study.

According to (4), for Gaussian kernel $n(x,y) = k(x,y)$, but for polynomial kernel $n(x,y) \neq k(x,y)$. In the following section normalized kernels are used to develop the kernel based FCM algorithm.

3. NKFCM: an unsupervised kernel based fuzzy clustering algorithm with kernel normalization

3.1 *FCM clustering algorithm*

A pioneering work on fuzzy cluster analysis was due to Ruspini [20], in which the first fuzzy clustering criterion function was proposed. However, it was the FCM or fuzzy ISODATA algorithm that established clearly the relevance of the theory of fuzzy sets to cluster analysis and pattern recognition [11].

In the FCM, an unsupervised fuzzy clustering algorithm partitions a given finite data set $X = \{x_1, \cdots, x_T\} \subset \Re^P$ into *c* fuzzy subsets. This fuzzy *c*-partition can be described by a partition matrix *U* as follows:

$$M_{fcT} = \left\{ U \in [0,1]^{c \cdot T} \ | \sum_{k=1}^{c} u_{kj} = 1 \ \forall j; \ 0 < \sum_{j=1}^{T} u_{kj} < T \ \ \forall k \right\} \tag{8}$$

where the matrix element u_{kj} represents the membership of $x_j \in X$ in the *k*th cluster and is determined by minimizing the following objective function [11]:

$$J_m(U,V) = \sum_{k=1}^{c} \sum_{j=1}^{T} (u_{kj})^m d_{kj}^2 \tag{9}$$

where $U \in M_{fcT}$ is a fuzzy *c*-partition of *X*, $V = (V_1, \cdots, V_c) \in \Re^{c \cdot P}$, with $V_k \in \Re^P$ being the cluster center or prototype of the *k*th fuzzy subset, the distance d_{kj} is defined by

$$d_{kj}^2 = \left\| x_j - V_k \right\|^2 \tag{10}$$

and $m \in [1, \infty)$ is a weighting exponent. Criterion (9) is reduced to Dunn's functional by setting *m*=2 [21]. As $m \to 1$, the FCM solution becomes hard and converges in theory to a "generalized" hard *c*-means solution.

3.2 *NKFCM clustering algorithm*

Because the Euclidian distance is used in (10), the FCM algorithm shows the tendency to partition the data points in clusters of hyperspherical shape with an approximately equal number of data points in each cluster. Given a kernel function $k(x,y)$, by using a nonlinear mapping function $\Psi(\cdot) : \Re^P \to \Gamma$ in the form of (3), the clustering process can be carried out in feature space rather than input space and the above restrictions can be avoided. The criterion functional used in feature space is defined as

$$\begin{aligned} J_m(U, V^\Psi) &= \sum_{k=1}^{c} \sum_{j=1}^{T} (u_{kj})^m D_{kj}^2 \\ &= \sum_{k=1}^{c} \sum_{j=1}^{T} (u_{kj})^m \left\| \Psi(x_j) - V_k^\Psi \right\|^2 \end{aligned} \tag{11}$$

where $U \in M_{fcT}$ is a fuzzy *c*-partition of $\Psi(X) = \{ \Psi(x_1), \cdots, \Psi(x_T) \}$, which corresponds to the partition matrix on *X*, $V^\Psi = (V_1^\Psi, \cdots, V_c^\Psi)$ represent cluster prototypes in feature space. The optimal partition is obtained by

$$\arg\min_{U,V^{\Psi}} J_m(U,V^{\Psi}) \tag{12}$$

subject to the constraints in (8).

The expression of cluster prototypes V_k^{Ψ} in feature space can be derived as follows:

$$\begin{aligned}\frac{\partial J_m}{\partial V_k^{\Psi}} &= \sum_{j=1}^{T}(u_{kj})^m \frac{\partial D_{kj}^2}{\partial V_k^{\Psi}} \\ &= \sum_{j=1}^{T}(u_{kj})^m \frac{\partial}{\partial V_k^{\Psi}}\left[(\Psi(x_j)-V_k^{\Psi})^T\cdot(\Psi(x_j)-V_k^{\Psi})\right] \\ &= \sum_{j=1}^{T}(u_{kj})^m\left[-2(\Psi(x_j)-V_k^{\Psi})\right]\end{aligned} \tag{13}$$

Setting $\frac{\partial J_m}{\partial V_k^{\Psi}} = 0$ leads to

$$\begin{aligned}V_k^{\Psi} &= [\sum_{j=1}^{T}(u_{kj})^m \Psi(x_j)]/\sum_{j=1}^{T}(u_{kj})^m \\ &= \sum_{j=1}^{T}\tilde{u}_{kj}\Psi(x_j)\end{aligned} \tag{14}$$

where

$$\tilde{u}_{kj} = (u_{kj})^m / T_k = (u_{kj})^m / \sum_{j=1}^{T}(u_{kj})^m \tag{15}$$

Based on the expression (14) for V_k^{Ψ}, the distance D_{kj} in the objective function (11) can be reformulated as follows:

$$\begin{aligned}D_{kj}^2 &= \left(\Psi(x_j)-\sum_{l=1}^{T}\tilde{u}_{kl}\Psi(x_l)\right)^T\left(\Psi(x_j)-\sum_{i=1}^{T}\tilde{u}_{ki}\Psi(x_i)\right) \\ &= \Psi(x_j)^T\cdot\Psi(x_j) - 2\sum_{i=1}^{T}\tilde{u}_{ki}\Psi(x_i)^T\cdot\Psi(x_j) + \sum_{l=1}^{T}\sum_{i=1}^{T}\tilde{u}_{kl}\tilde{u}_{ki}\Psi(x_l)^T\cdot\Psi(x_i) \\ &= N_{jj} - 2\sum_{i=1}^{T}\tilde{u}_{ki}N_{ij} + \sum_{l=1}^{T}\sum_{i=1}^{T}\tilde{u}_{kl}\tilde{u}_{ki}N_{li} \\ &= 1 - 2\sum_{i=1}^{T}\tilde{u}_{ki}N_{ij} + \sum_{l=1}^{T}\sum_{i=1}^{T}\tilde{u}_{kl}\tilde{u}_{ki}N_{li}\end{aligned} \tag{16}$$

where N_{ij} is a symmetric $T \times T$ kernel matrix defined by

$$\begin{aligned}N_{ij} &= n(x_i,x_j) \\ &= \Psi(x_i)^T\cdot\Psi(x_j) \\ &= \frac{k(x_i,x_j)}{\sqrt{k(x_i,x_i)k(x_j,x_j)}}\end{aligned} \tag{17}$$

Hence, the criterion function (11) can be expressed as follows:

$$\begin{aligned}
J_m(U) &= \sum_{k=1}^{c}\sum_{j=1}^{T}(u_{kj})^m D_{kj}^2 \\
&= \sum_{k=1}^{c}\sum_{j=1}^{T}(u_{kj})^m - 2\sum_{k=1}^{c}\sum_{j=1}^{T}\sum_{i=1}^{T}(u_{kj})^m \widetilde{u}_{ki} N_{ij} + \sum_{k=1}^{c}\sum_{j=1}^{T}\sum_{l=1}^{T}\sum_{i=1}^{T}(u_{kj})^m \widetilde{u}_{kl}\widetilde{u}_{ki} N_{li} \\
&= \sum_{k=1}^{c}\sum_{j=1}^{T}(u_{kj})^m - 2\sum_{k=1}^{c} T_k \left(\sum_{j=1}^{T}\sum_{i=1}^{T}\widetilde{u}_{kj}\widetilde{u}_{ki} N_{ij}\right) + \sum_{k=1}^{c}\left(\sum_{j=1}^{T}(u_{kj})^m\right)\sum_{l=1}^{T}\sum_{i=1}^{T}\widetilde{u}_{kl}\widetilde{u}_{ki} N_{li} \\
&= \sum_{k=1}^{c}\sum_{j=1}^{T}(u_{kj})^m - \sum_{k=1}^{c} T_k \left(\sum_{j=1}^{T}\sum_{i=1}^{T}\widetilde{u}_{kj}\widetilde{u}_{ki} N_{ij}\right) \\
&= \sum_{k=1}^{c}\sum_{j=1}^{T}(u_{kj})^m - \sum_{k=1}^{c} \frac{1}{T_k}\left(\sum_{j=1}^{T}\sum_{i=1}^{T}(u_{kj})^m (u_{ki})^m N_{ij}\right)
\end{aligned} \tag{18}$$

where $T_k = \sum_{j=1}^{T}(u_{kj})^m$. From the above criterion function, it is clear that based on the kernel expression the clustering in feature space makes no implicit assumption concerning the shape of the clusters such as hyper-spherical or hyper-ellipsoidal structure in input space.

3.2.1 *Optimal partitioning matrix*

The first problem in developing the NKFCM algorithm is how to optimize the partitioning matrix *U* by minimizing (18) subject to the constraints in (8). Defining the following Lagrangian:

$$L(U,\beta) = J_m + \sum_{j=1}^{T}\beta_j\Big(\sum_{k=1}^{c}u_{kj} - 1\Big) \tag{19}$$

and setting the Lagrangian's gradient to zero, we obtain

$$\frac{\partial L(U,\beta)}{\partial \beta_j} = \sum_{k=1}^{c}u_{kj} - 1 = 0 \tag{20}$$

$$\frac{\partial L(U,\beta)}{\partial u_{kj}} = m(u_{kj})^{m-1}\left(1 - \frac{2}{T_k}\sum_{i=1}^{T}(u_{ki})^m N_{ij} + \frac{1}{T_k^2}\sum_{l=1}^{T}\sum_{i=1}^{T}(u_{kl})^m (u_{ki})^m N_{li}\right) + \beta_j = 0 \tag{21}$$

Let

$$\rho_{kj} =: 1 - \frac{2}{T_k}\sum_{i=1}^{T}(u_{ki})^m N_{ij} + \frac{1}{T_k^2}\sum_{l=1}^{T}\sum_{i=1}^{T}(u_{kl})^m (u_{ki})^m N_{li} \tag{22}$$

From (21), we get

$$u_{kj} = \left(-\frac{\beta_j}{m\rho_{kj}}\right)^{\frac{1}{m-1}} \tag{23}$$

Combination of (20) and (23) leads to:

$$\left(-\beta_j\right)^{\frac{1}{m-1}} = \frac{1}{\sum_{k=1}^{c}\left(m\rho_{kj}\right)^{\frac{1}{1-m}}} \tag{24}$$

Hence

$$u_{kj} = \frac{\left(\rho_{kj}\right)^{\frac{1}{1-m}}}{\sum_{k=1}^{c}\left(\rho_{kj}\right)^{\frac{1}{1-m}}} \tag{25}$$

When $\rho_{kj} = 0$, i.e., the sample data x_j is located at the core center of the fuzzy set for the *k*th cluster, special care is needed. Firstly, the data classes can be divided into two groups I_j and $\tilde{I}_j$ as follows:

$$\forall\ 1 \le j \le T, \quad I_j =: \{k \mid 1 \le k \le c; \rho_{kj} = 0\}, \quad \tilde{I}_j =: \{1, \cdots, c\} - I_j \tag{26}$$

If $I_j \ne \phi$, then set $u_{kj} = 0$ for $k \in \tilde{I}_j$ and make $\sum_{k \in I_j} u_{kj} = 1$ for $k \in I_j$.

3.2.2 *Cluster prototypes in input space*
Another problem in developing the NKFCM algorithm is how to obtain the cluster prototypes in input space. After an optimal partition *U* that minimizes the criterion functional (18) is obtained, the cluster prototypes V_k^{Ψ} can be expressed as expansions of mapped patterns. However, as discussed before it is difficult to obtain explicit expressions for the mapped patterns, and even if explicit expressions are available it is not guaranteed that there exist a preimage pattern v_k in input space such that $\Psi(v_k) = V_k^{\Psi}$ since the mapping function $\Psi(x_i)$ is nonlinear. The problem about cluster prototypes in kernel based clustering methods is far from being addressed in the literature [16]. This paper proposes a new method to calculate cluster prototypes in input space in terms of kernel function rather than mapping function Ψ. Specifically speaking, approximate prototypes in input space can be obtained by minimizing the following functional:

$$W(v_k) = \left\| \Psi(v_k) - V_k^{\Psi} \right\|^2 \tag{27}$$

By replacing (14) for V_k^{Ψ}, $W(v_k)$ can be expressed by expansions of inner products of the mapping functions, and the inner products can be replaced by kernel functions. Thus, $W(v_k)$ can be reformulated as follows:

$$\begin{aligned} W(v_k) &= \sum_{l=1}^{T}\sum_{i=1}^{T} \tilde{u}_{kl}\tilde{u}_{ki} n(x_l, x_i) - 2\sum_{j=1}^{T} \tilde{u}_{kj} n(x_j, v_k) + n(v_k, v_k) \\ &= \sum_{l=1}^{T}\sum_{i=1}^{T} \tilde{u}_{kl}\tilde{u}_{ki} n(x_l, x_i) - 2\sum_{j=1}^{T} \tilde{u}_{kj} n(x_j, v_k) + 1 \end{aligned} \tag{28}$$

As a matter of fact, the minimization of (28) corresponds to minimizing the distance between V_k^{Ψ} and the orthogonal projection of V_k^{Ψ} onto span$(\Psi(v_k))$, which is equivalent to maximizing the following term:

$$\begin{aligned} \frac{\left(V_k^{\Psi} \cdot \Psi(v_k)\right)^2}{\left(\Psi(v_k) \cdot \Psi(v_k)\right)} &= \left(V_k^{\Psi} \cdot \Psi(v_k)\right)^2 \\ &= \left(\sum_{j=1}^{T} \tilde{u}_{kj} n(x_j, v_k) \right)^2 \end{aligned} \tag{29}$$

Scholkopf et al approximated a preimage pattern in input space for a given mapped pattern in feature space in terms of (29) [16] without consideration of kernel normalization by using $\Phi(\cdot)$ instead of $\Psi(\cdot)$. Obviously, the optimal cluster prototypes in input space is the solution of the following equation

$$\sum_{j=1}^{T} \tilde{u}_{kj} \frac{\partial n(x_j, v_k)}{\partial v_k} = 0 \tag{30}$$

Hence, by applying (30), for Gaussian kernels, the approximated cluster prototypes in input space is obtained as follows,

$$v_k = \frac{\sum_{j=1}^{T} (u_{kj})^m n(x_j, v_k) x_j}{\sum_{j=1}^{T} (u_{kj})^m n(x_j, v_k)} \tag{31}$$

and the approximated cluster prototypes in input space for normalized polynomial kernels can be expressed as

$$\begin{aligned} v_k &= \frac{\sum_{j=1}^{T} \tilde{u}_{kj} \big((x_j \cdot v_k) + \theta\big)^{d-1} x_j}{\big((v_k \cdot v_k) + \theta\big)^{d-1}} \\ &= \frac{\sum_{j=1}^{T} (u_{kj})^m \big((x_j \cdot v_k) + \theta\big)^{d-1} x_j}{T_k \cdot \big((v_k \cdot v_k) + \theta\big)^{d-1}} \end{aligned} \tag{32}$$

3.3 *Picard iteration in the NKFCM algorithm*

In the implementation of the NKFCM algorithm, the following steps should be included:

Step 1. Set the number of clusters c ($1<c<T$), the weighting exponent $m \in (1, \infty)$, and the error threshold $\varepsilon_U \geq 0$. Initialize partition matrix $U^{(0)} \in M_{fcT}$. Set t=0.

Step 2. Choose a kernel function: Gaussian kernel or polynomial kernel, and perform the kernel normalization.

Step 3. Set t=:t+1. Update the fuzzy partition matrix $U^{(t)}$ in terms of (25).

Step 4. If $\left\| U^{(t)} - U^{(t-1)} \right\| < \varepsilon_U$, go to Step 5; otherwise go to Step 3.

Step 5. Calculate the cluster prototypes in input space:

Step 5.1 For optimal partition matrix $U^{(*)}$, set error threshold $\varepsilon_V \geq 0$. Initialize cluster prototypes $V^{(0)} = (v_1^{(0)}, \cdots, v_c^{(0)}) \in \Re^{Pc}$. Set r=0.

Step 5.2 Set r=:r+1. Update the cluster prototypes $V^{(r)}$ using (31) for Gaussian kernels or (32) for normalized polynomial kernels.

Step 5.3 If $\left\| V^{(r)} - V^{(r-1)} \right\| < \varepsilon_V$, stop; otherwise go to Step 5.2.

4. Experimental results

Three examples of unsupervised data clustering are given in this section to demonstrate how the proposed fuzzy clustering algorithm works in different scenarios.

In the first example, the NKFCM algorithm is applied to the Ringnorm dataset available at DELVE repository (http://www.cs.toronto.edu/~delve/data/ringnorm). This dataset consists of 2 classes of 7400 samples, each containing 20 attributes. The data samples are drawn from multivariate normal distributions. Breiman reported that the theoretically expected error rate on this data set is 1.3% [22]. The Ringnorm data is labeled. The labels are not used in the following clustering process, but used for calculating the clustering error rate $E(U)$ to evaluate the performance of the NKFCM algorithm. The clustering error rate is defined by [11]

$$E(U) = \frac{1}{2T} \sum_{k=1}^{c} \sum_{j=1}^{T} (\hat{u}_{kj} - \bar{u}_{kj})^2 \tag{33}$$

where $\bar{u}_{kj}$ (=0 or 1) are labels from the dataset and $\hat{u}_{kj}$ form a hard partition that is the closest to fuzzy partition U, i.e.,

$$\hat{u}_{kj} = \begin{cases} 1 & \textit{if } u_{kj} = \max_l (u_{lj}) \\ 0 & \textit{otherwise} \end{cases} \tag{34}$$

In order to examine the influence of the kernel parameters on the clustering performance, a group of parameter values for Gaussian kernel and polynomial kernel are evaluated separately in the experiment. The experimental results are illustrated in Fig.1 and Fig.2. It can be seen that both the width of the Gaussian kernel and the offset and exponential parameters of the polynomial kernel have a significant effect on the outcomes of the clustering. By choosing appropriate values for kernel functions, the NKFCM algorithm exhibits striking performance on this dataset. Set $\sigma = 6.5$ for Gaussian kernel and *m*=2, the NKFCM algorithm leads to 99 misclassifications among 7400 samples with an error rate of 1.34%, which is very close to the theoretically expected value 1.3% [22]. The NKFCM algorithm with normalized polynomial kernel also achieves a promising performance at $\theta = 40$, $d = 4$ and *m*=2 with an error rate of 2.62%. As a comparison, Fig.3 shows the results obtained by kernel based FCM algorithm without consideration of the kernel normalization for polynomial kernel (7), in which the optimal performance, an error rate of 4%, is achieved by setting $\theta = 4$, $d = 2$ and *m*=2. This indicates that by the proposed kernel normalization the performance of the NKFCM algorithm has been improved. This is reasonable, because due to kernel normalization some bizarrerie within the original patterns could be relieved during clustering.

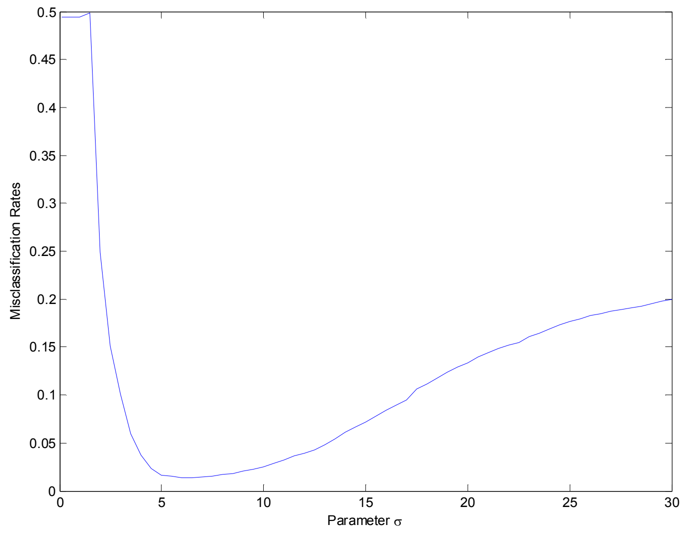

Fig.1 Error rates obtained by NKFCM with Gaussian kernel vs kernel parameter σ

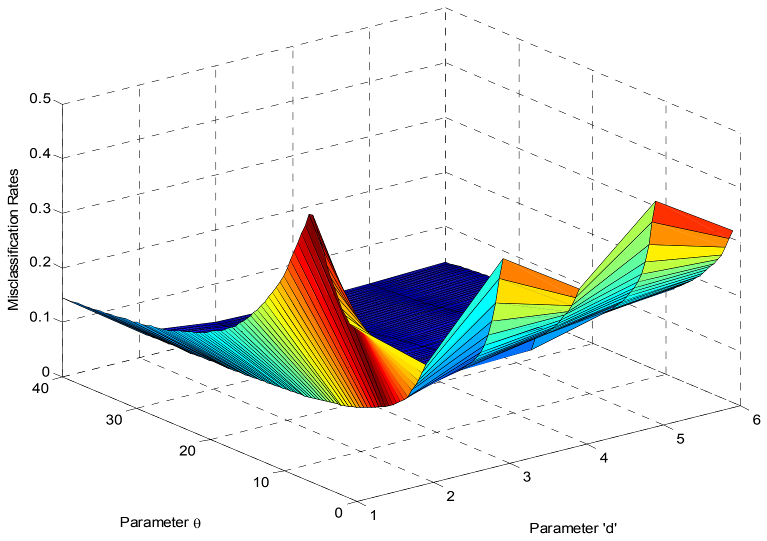

Fig.2 Error rates obtained by NKFCM with normalized polynomial kernel vs kernel parameters θ and d

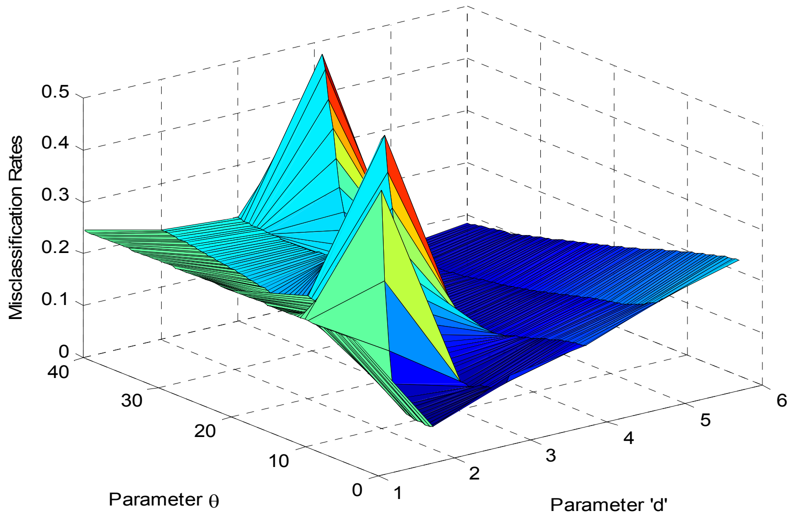

Fig.3 Error rates obtained by kernel based FCM without consideration of kernel normalization for polynomial kernel vs kernel parameters θ and d

In order to compare with other methods, Table 1 shows the error rates of the NKFCM method, the decision trees CART [22][23] and the FCM algorithm on Ringnorm dataset. It should be noted that the error rate from CART was obtained in a reduced Ringnorm dataset with only 300 samples [22]. The advantages of the proposed kernel normalization procedure and the NKFCM algorithm are very obvious from Table 1. However, in comparison with non-kernel based methods such as FCM algorithm, the drawback of the NKFCM algorithm is also obvious due to the high computational load, although this drawback commonly exists in most kernel based methods. Fig.4, Fig.5, and Fig6 show the time costs by the NKFCM algorithm with Gaussian kernel (σ=6.5, m=2), the NKFCM algorithm with normalized polynomial kernel (d=4, θ=40, m=2) and the FCM algorithm (m=2) separately against the variation of the number of used samples. It should be noted that for Gaussian kernel the time cost is dramatically increased when the number of used samples is greater than 6300. This is caused by the use of hard disk as virtual memory. Similarly, for normalized polynomial kernel this dramatic increase happens when the number of used samples is greater than7200.

Table 1. Performance comparison on Ringnorm dataset

Methods	*Error rates*	*No. of samples used*
NKFCM (m=2) with Gaussian kernel (σ=6.5)	1.34%	7400 samples
NKFCM (m=2) with normalized polynomial kernel $n(x,y)$ (d=4, θ=40)	2.62%	7400 samples
Kernel based FCM (m=2) with polynomial kernel $k(x,y)$ (d=2, θ=4)	4%	7400 samples
FCM (m=2)	24.01%	7400 samples
CART[22][23]	21.4%	300 samples

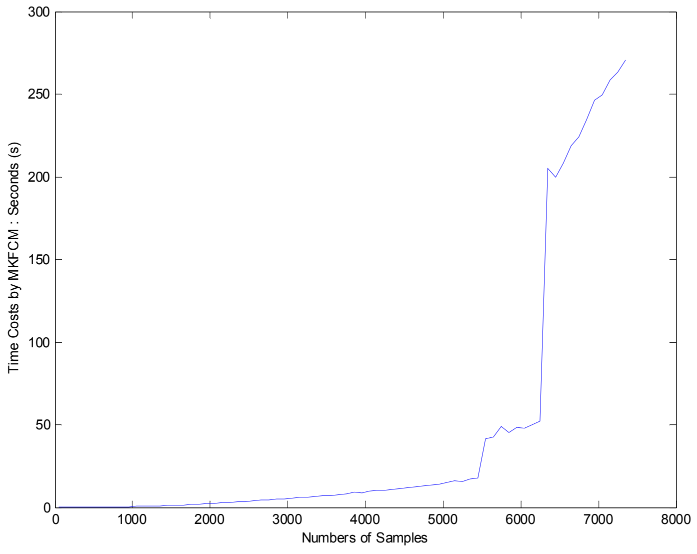

Fig.4 Time costs by NKFCM with Gaussian kernel (σ =6.5, *m*=2)

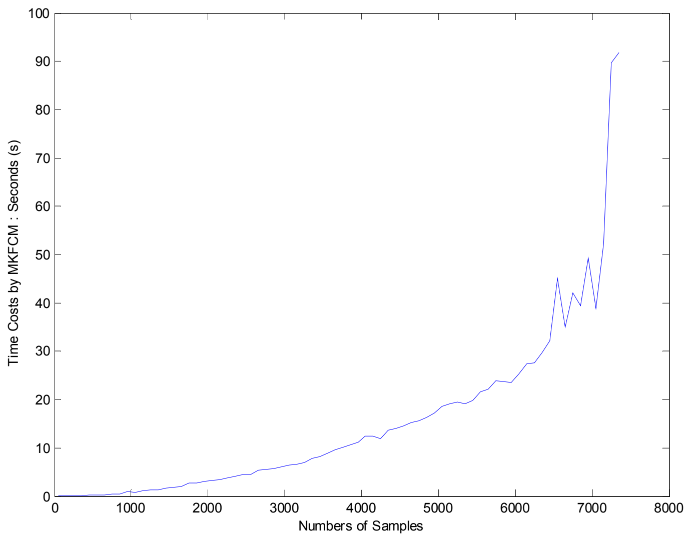

Fig.5 Time costs by NKFCM with normalized polynomial kernel (*d*=4, θ =40, *m*=2)

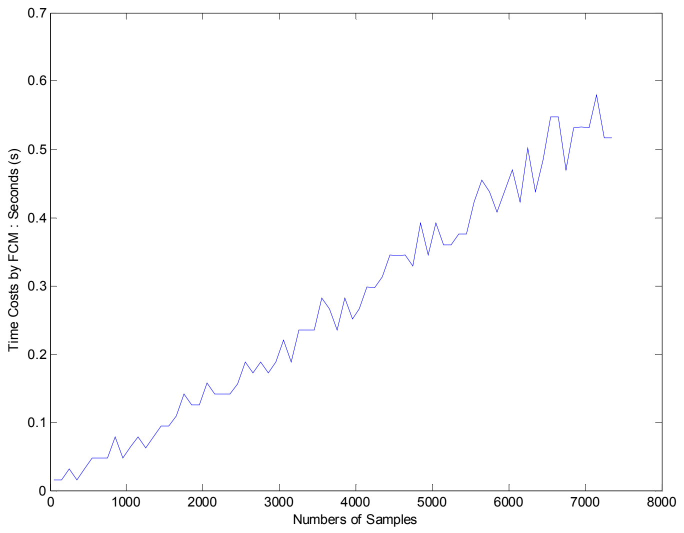

Fig.6 Time costs by FCM (m=2)

The purpose of the second example is to test the performance of the proposed NKFCM in detecting non-hyperspherical clusters, in which a synthetic dataset with a cross structure is used, as shown in Fig.7. The NKFCM algorithm with Gaussian kernel finds out two line-shaped clusters, as illustrated in Fig.8. Moreover, the NKFCM algorithm with the normalized polynomial kernel obtains parabola-shaped clusters, as shown in Fig.9. Fig.10 indicates that the FCM algorithm failed to detect the line-shape clusters due to its favor of hyperspherical or hyperelliptical clusters.

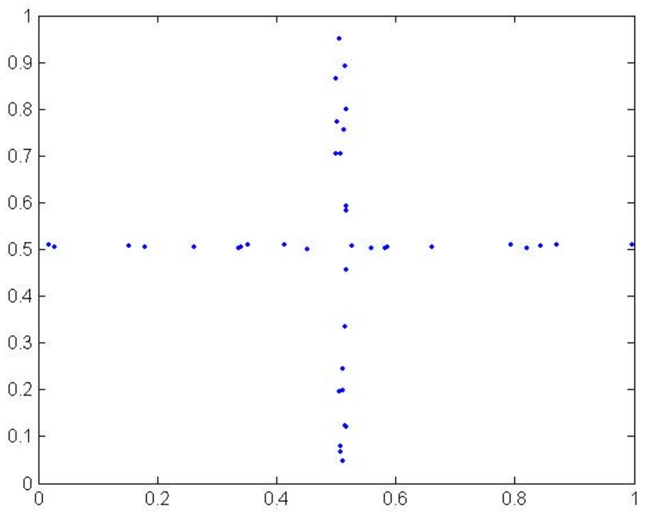

Fig. 7 Cross dataset

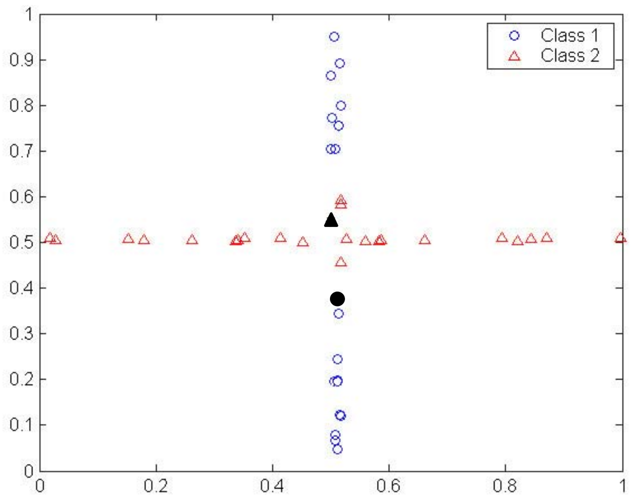

Fig. 8 Clustering results by NKFCM: *m*=2 and Gaussian kernel with σ =0.058.
"●" and "▲" : the prototypes of clusters.

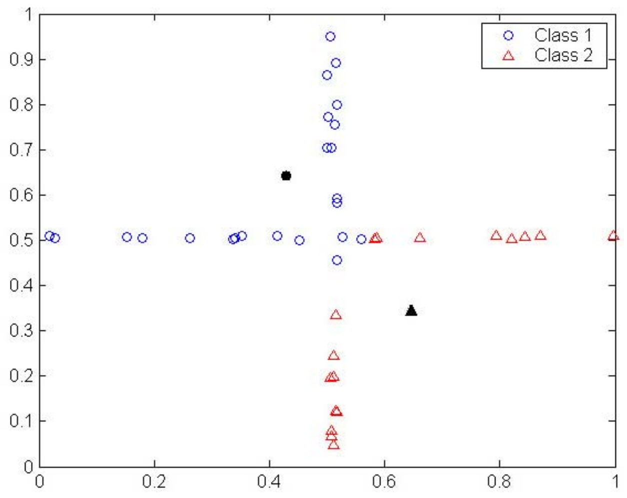

Fig. 9 Clustering results by NKFCM: *m*=2 and normalized polynomial kernel with *d*=2, θ =0.01.
"●" and "▲" : the prototypes of clusters.

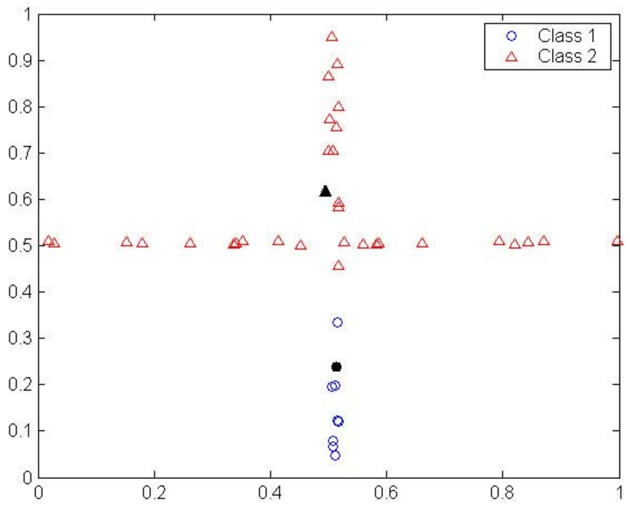

Fig. 10 Clustering results by FCM: *m*=2 and *c*=2. "●" and "▲": the prototypes of clusters.

In the third example, the NKFCM algorithm is evaluated on the well-known Iris dataset [24]. This dataset contains three classes of data: *Iris setosa*, *Iris versicolor,* and *Iris virginica*, each data point containing four features: *Sepal length, Sepal width, Petal length,* and *Petal width* measured in millimeters. There are 50 samples for each of the three classes, which are shown in Fig.11 in the *Sepal length* vs. *Sepal width* plane. It can be seen that one of the classes is linearly separable from the other two and the remaining two classes are significantly overlapped. Similarly, the clustering error rate described by (33) is used to evaluate the performance of the NKFCM algorithm. Fig.12 shows the clustering result obtained by the NKFCM algorithm using Gaussian kernel with m=2 and $\sigma = 12$ in the *Sepal length* vs. *Sepal width* plane. There are 10 misclassifications and the clustering error rate is 6.67%. As a comparison to the NKFCM algorithm, the performances of several other methods on the same dataset are given in Table 2. The error rate of the FCM is 10.67%, corresponding to 16 misclassifications. The clustering result obtained by the FCM algorithm is shown in Fig.13. The SVC algorithm [5] used principal components rather than the original Iris data. When four principal components were considered, the number of misclassifications of the SVC on the Iris dataset is 14, corresponding to an error rate of 9.33%. The SPC algorithm by Blatt et al misclassified 15 data points when applied to the original Iris dataset [25]. From Table 2, it is clear that the NKFCM algorithm achieves the best performance.

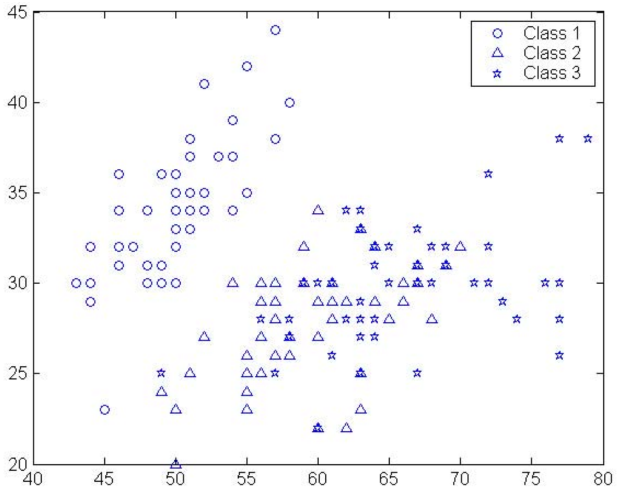

Fig.11 Iris data in the *Sepal length vs Sepal width* plane. Class 1-Iris Setosa, Class 2-Iris Versicolor, and Class 3-Iris Virginica

Table 2. Comparison of several algorithms on Iris dataset

Method	*No. of misclassifications*	*Error rate*
NKFCM with Gaussian kernel (m=2, $\sigma = 12$)	10	6.67%
FCM (m=2)	16	10.67%
SVC [5]	14	9.33%
SPC [25]	15	10%

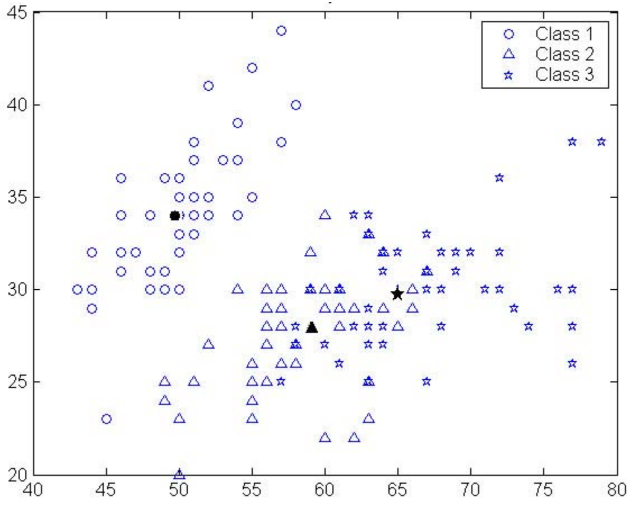

Fig.12 Clusters obtained by NKFCM with m=2 and $\sigma = 12$. Class 1-Iris Setosa, Class 2-Iris Versicolor, and Class 3-Iris Virginica. "●", "▲" and "★": the prototypes of clusters.

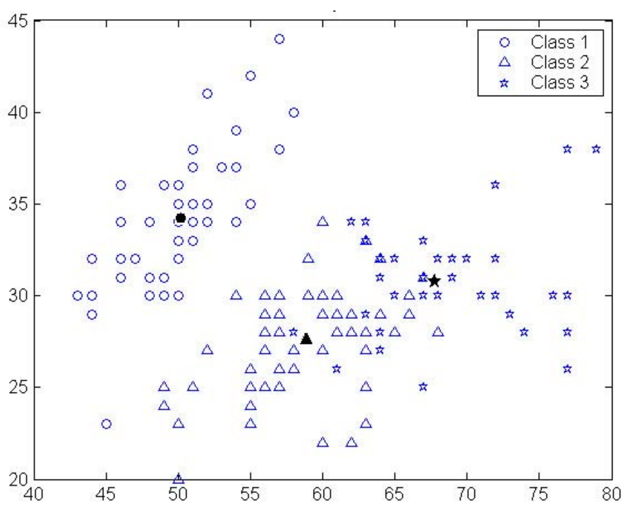

Fig.13 Clusters obtained by FCM with *m*=2. Class 1-Iris Setosa, Class 2-Iris Versicolor, and Class 3-Iris Virginica. "●", "▲" and "★" : the prototypes of clusters.

5. Conclusions and discussions

In this paper, a new kernel normalization procedure is suggested, based on which the normalized Mercer kernel possesses clear geometric interpretation and some good properties. An unsupervised clustering algorithm, NKFCM, is also proposed. The NKFCM algorithm makes no assumptions about the shape of clusters within data samples, it thus possesses strong adaptability to data cluster structures. In virtue of normalized kernel transformation from input space into high-dimensional feature space, the opportunity of linearly separating the mapped patterns by the NKFCM algorithm increases, which has been supported by the experimental results. However, it becomes problematic to calculate the prototypes of clusters in input space in Mercer kernel based clustering methods. In the NKFCM algorithm, a method for approximating the prototypes of clusters in input space has been developed. Experimental results have demonstrated the promising performance of the NKFCM algorithm with kernel normalization procedure.

From the experiments it is also found that the kernel parameters have great influence on the clustering performance. Kernel functions with different parameters give rise to different results. Hence, how to choose an appropriate kernel parameter is a problem that has not been solved well in this paper. As a matter of fact, this is an open problem in most kernel based algorithms. A possible way to attack this open problem is to optimize kernel parameters in terms of a criterion for kernel parameter selection. Another problem is how to speed up the kernel matrix computation and lighten the requirements on computer memory, which is also a common demand in most kernel based algorithms. These problems will receive much attention in our future research.

References

[1] V. N. Vapnik, *The Nature of Statistical Learning Theory*. (New York:Springer-Verlag, 1995).

[2] B. Schölkopf, A. J. Smola, and K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, *Neural Computation*, 10(1998)1299–1319.

[3] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, Fisher discriminant analysis with kernels, in *Neural Networks for Signal Processing IX*, Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds. (Piscataway, NJ: IEEE, 1999)41–48.

[4] A. Ruiz, and P. E. López-de-Teruel, Nonlinear kernel-based statistical pattern analysis, *IEEE Trans. on Neural Networks*, 12(2001)16-32.

[5] A.Ben-Hur, D. Horn, H. T. Siegelmann, V. Vapnik, Support vector clustering, *Journal of Machine Learning Research,* 2 (2001) 125-137

[6] M. M. Van Hulle, Kernel-based equiprobabilistic topographic map formation, *Neural Computation*, 10(1998)1847–1871.

[7] M. Girolami, Mercer kernel-based clustering in feature space, *IEEE Trans. on Neural Networks*, 13(2002)780-784.

[8] C.-F. Lin and S.-D. Wang, Fuzzy support vector machines, *IEEE Trans. on Neural Networks*, 13(2002)464-471.

[9] D.-Q. Zhang and S.-C. Chen, Clustering incomplete data using kernel-based fuzzy c-means, *Neural Processing Letters*, 18(2003) 155-162.

[10] J.-H. Chiang and P.-Y. Hao, A new kernel-based fuzzy clustering approach: support vector clustering with cell growing, *IEEE Trans. on Fuzzy Systems*, 11(2003)518-527.

[11] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, (Plenum Press, New York,1981)

[12] D. E. Gustafson and W. Kessel, Fuzzy clustering with a fuzzy covariance matrix, in *Proc. IEEE-CDC*, Voll.2(K. S. Fu ed), (IEEE Press, Piscataway, new Jersey, 1979)761-766.

[13] J. C. Bezdek and I. Anderson, An application of the *c*-varieties clustering algorithm to polygonal curve fitting, *IEEE Trans. on Systems, Man, and Cybernetics,* 15(1985)637-641.

[14] C. Jerome, B. Noel, and H. Michel, A new fuzzy clustering technique based on pdf estimation, *Proceedings of Information Processing and Managing of Uncertainty (IPMU'2002) 225-232.*

[15] F. Hoeppner, Fuzzy shell clustering algorithms in image processing: fuzzy C-rectangular and 2-rectangular shells, *IEEE Trans. on Fuzzy Systems*, 5(1997)599-613.

[16] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Ratsch, and A. J. Smola, Input space versus feature space in kernel based methods, *IEEE Trans. on Neural Networks*, 10(1999)1000–1017.

[17] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, An introduction to kernel-based learning algorithms, *IEEE Trans. on Neural Networks*, 12(2001)181-201.

[18] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, *Philos. Trans. Roy. Soc. London*, A209(1909)415–446.

[19] D. M. J. Tax, and R.P.W. Duin, Support vector domain description, *Pattern Recognition Letters,* 20(1999)1191-1199.

[20] E. Ruspini, Numerical methods for fuzzy clustering, *Information Science*, 2(1970) 319-350.

[21] J.C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated cluster, *Journal of Cybernetics*, 3(1973)32-57.

[22] L. Breiman, Bias, variance and arcing classifiers, *Tech. Report 460*, Statistics department. University of California, USA. April 1996.

[23] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees,* (Chapman & Hall / CRC, 1984)

[24] R.A. Fisher, The use of multiple measurements in taxonomic problems, *Annals of Eugenics,* 7(1936)179–188.

[25] M. Blatt, S. Wiseman, and E. Domany, Data clustering magnet, *Neural Computation,* 9(1997)1805–1842.