Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

Fuzzy c-Means Classifier with Deterministic Initialization and Missing Value Imputation

Hidetomo Ichihashi, Katsuhiro Honda, Akira Notsu, and Takafumi Yagi
Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531 Japan

***Abstract*— A fuzzy c-means (FCM) classifier derived from a generalized FCM clustering is proposed. The clssifier design is based on FCM. The classifier is not initialized with random numbers, hence being deterministic. The parameters are optimized by cross validation (CV) protocol and golden section search method. A method for dealing with missing values without eliminating them but with estimating them is also proposed. Instead of using the terminology "conditional expectation", the imputation is done by the least square method of Mahalanobis distances between the datum with missing values and cluster centers. The FCM classifier outperforms well established methods such as support vector machine, k-nearest neighbor and Gaussian mixture classifiers for datasets with and without missing values.**

I. Introduction

The standard fuzzy c-means (FCM) clustering objective function [1] is generalized a little further, and iterative rules for clustering is derived. The clustering algorithm, which is derived from the generalized FCM and the iteratively reweighted least square (IRLS) technique [2] is applied to a post-supervised classifier. Cluster memberships are defined by a function of Mahalanobis distances between data vectors and cluster centroids. The algorithm is called FCM classifier [3], [4], [5]. In the post-supervised design, the clustering is done on a per class basis and is implemented by using the data from one class at a time. When working with the data class by class, the prototypes that are found for each labeled class already have the assigned physical labels.

High performance classifiers usually have parameters to be selected. For example, support vector machine (SVM) [6], [7] has the regularization parameter and kernel parameters. After selecting the best parameters by some procedure, we use the parameters to train the whole training set, and then test new unseen data. For making our FCM classifier deterministic, this paper proposes a way of determining initial cluster centroids based on principal component (PC) basis vectors. No random initializatin is used and this enables us to evaluate the classifier performance by a single run of cross varidation test with a default partition of benchmark data. A parameter optimization procedure with grid search and golden section search is proposed.

FCM classifier outperforms well-established presupervised methods, i.e., SVM, decision tree approach C4.5 [8], k-nearest neighbor classier (k-NN), learning vector quantization (LVQ) [9], [10] and Gausian mixture classifier (GMC) [11], [12] on the benchmark data sets from the UCI ML repository [13].

Missing values are common in many real world data sets. The interest in dealing with missing values has continued with the applications to data mining and microarrays [14]. These applications include supervised classification as well as unsupervised classification (clustering).

Usually entire incomplete data samples with missing values are eliminated in preprocessing (the case deletion method). Other well known methods are the mean imputation, median imputation and nearest neighbor imputation [15] procedure. The nearest neighbor algorithm searches through all the dataset looking for the most similar instances. This is a very time consuming process and it can be very critical in data mining where large databases are analyzed.

In the multiple imputation method the missing values in a feature are filled in with values drawn randomly (with replacement) from a fitted distribution for that feature. This procedure is repeated a number of times [16].

In the local principal component analysis (PCA) with clustering [17], [18], not the entire data samples but only the missing values are ignored by multiplying "0" weights over the corresponding reconstruction errors.

Maximum likelihood procedures that use variants of the Expectation-Maximization algorithm can handle parameter estimation in the presence of missing data. These methods are generally superior to case deletion methods, because they utilize all the observed data. However, they suffer from a strict assumption of a model distribution for the variables, such as a multivariate normal model, which has a high sensitivity to outliers.

This paper proposes an approach to clustering and classification without eliminating or ignoring missing values but with estimating the values. Instead of computing conditional expectation using a probability distribution, the estimation is done by the least squares method of Mahalanobis distances between the datum with missing values and cluster centroids.

We carry out experiments with benchmark datasets to evaluate the effect on the classification error rate of the methods for dealing with missing values: the proposed, EM algorithm of GMC, k-NN with nearest neighbor imputation procedure.

The paper is organized as follows. Section II gives a brief description of the generalized FCM clustering and the classifier design based on IRLS. A parameter optimization procedure with grid search and golden section search is proposed in Section III. Our imputation methods by minimizing Mahalanobis distances between data and cluster centers will be described in Section IV. Section V provides the results of numerical experiments. Section VI concludes the paper.

II. Generalized FCM Clustering and Classifier

The clustering is used as an unsupervised phase of the classifier design. FCM clustering partitions data set by introducing memberships to fuzzy clusters. Let r dimensional vector $\boldsymbol{v}_i$ denote prototype parameter (i.e., cluster centroid). u_{ik} denotes the membership of k-th object data $\boldsymbol{x}_k \in \mathcal{R}^r$ to i-th cluster.

We first summarize the three kinds of objective functions, i.e., the standard, entropy-based, and quadratic-term-based fuzzy c-means. The objective function of the standard method is:

$$J_{\rm fcm} = \sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^{\lambda} d_{ik}^2, \quad (\lambda > 1). \tag{1}$$

c denotes the number of clusters. d_{ik}^2 denotes the squared distance between data vector $\boldsymbol{x}_k$ and centroid vector $\boldsymbol{v}_i$, so the standard FCM objective function is the weighted sum of squared distances. Following objective function is used for the entropy-based method.

$$J_{\rm efc} = \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik} d_{ik}^2 + \eta \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik} \log u_{ki}, \quad (\eta > 0). \tag{2}$$

The objective function of the quadratic-term-based method [19] is:

$$J_{\rm qfc} = \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik} d_{ik}^2 + \eta \sum_{i=1}^{c}\sum_{k=1}^{n} (u_{ik})^2 \quad (\eta > 0). \tag{3}$$

λ and η serves as fuzzifiers and the larger the λ and η the fuzzier the partition. From the above comparison we can generalize the standard objective function a little further as:

$$J_{\rm gfc} \;=\; \sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ki})^{\lambda} d_{ik}^2 + \eta \sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^{\lambda}, \tag{4}$$

where $\eta > 0, \lambda > 1$. From the necessary condition of optimality under the condition that $\sum_{i=1}^{c} u_{ik} = 1$, we have

$$u_{ik} = \left[\sum_{j=1}^{c}\left(\frac{\eta + d_{ik}^2}{\eta + d_{jk}^2}\right)^{\frac{1}{\lambda-1}}\right]^{-1}, \tag{5}$$

$$\boldsymbol{v}_i = \frac{\sum_{k=1}^{n}(u_{ik})^{\lambda}\boldsymbol{x}_k}{\sum_{k=1}^{n}(u_{ik})^{\lambda}}. \tag{6}$$

where, $\boldsymbol{x}_k$ is the data vector of k-th sample. For more details of the generalized FCM clustering, see [20].

The objective function (2) includes the entropy term and is the only case where covariance matrices (A_i) can be taken into account. Although Gustafson and Kessel's modified FCM [21] is derived from an objective function with fuzzifier λ, we need to specify the values of determinant $|A_i|$ for all i. In order to deal with covariance structure within the scope of fuzzy c-means clustering, we need some simplifications based on the IRLS technique. Runkler and Bezdek's [22] fuzzy clustering scheme called alternating cluster estimation (ACE) is this kind of simplification.

Now we consider to deploy a technique from the robust M-estimation [2], [23]. The M-estimators try to reduce the effect of outliers by replacing the squared residuals with ρ-function, which is chosen to be less increasing than square. Instead of solving directly this problem, we can implement it as the IRLS. While the IRLS approach does not guarantee the convergence to a global minimum, experimental results have shown reasonable convergence points. If one is concerned about local minima, the algorithm can be run multiple times with different initial conditions.

Let the objective function of the IRLS-FCM be

$$J_{\rm ifc} = \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}\left(d_{ik}^2 + \log|A_i|\right), \tag{7}$$

where

$$d_{ik}^2 = (\boldsymbol{x}_k - \boldsymbol{v}_i)^{\top} A_i^{-1} (\boldsymbol{x}_k - \boldsymbol{v}_i) \tag{8}$$

is Mahalanobis distance from $\boldsymbol{x}_k$ to i-th cluster centroid. A_i is a covariance matrix of data samples of the i-th cluster, which is derived from (7) as:

$$A_i = \frac{\sum_{k=1}^{n} u_{ik}(\boldsymbol{x}_k - \boldsymbol{v}_i)(\boldsymbol{x}_k - \boldsymbol{v}_i)^{\top}}{\sum_{k=1}^{n} u_{ik}}, \tag{9}$$

and $\boldsymbol{v}_i$ is derived as:

$$\boldsymbol{v}_i = \frac{\sum_{k=1}^{n} u_{ik}\boldsymbol{x}_k}{\sum_{k=1}^{n} u_{ik}}. \tag{10}$$

Membership function u_{ik} is not derived from the objective function and is a pre-defined weight function. We confine our discussion to the function

$$u_{ik}^{*} \;=\; \frac{\pi_i |A_i|^{-1/\gamma}}{(\eta + d_{ik}^2/0.1)^{1/\lambda}}. \tag{11}$$

To facilitate competitive movements of cluster centroids, we need to define the weight function to be normalized as:

$$u_{ik} = \frac{u_{ik}^{*}}{\sum_{l=1}^{c} u_{lk}^{*}}, \tag{12}$$

then u_{ik} is written as:

$$u_{ik} = \pi_i |A_i|^{-1/\gamma}\left[\sum_{j=1}^{c}\left(\frac{\eta + d_{ik}^2/0.1}{\eta + d_{jk}^2/0.1}\right)^{\frac{1}{\lambda}} \pi_j |A_j|^{-1/\gamma}\right]^{-1}. \tag{13}$$

$$\pi_i = \frac{\sum_{k=1}^{n} u_{ik}}{\sum_{j=1}^{c}\sum_{k=1}^{n} u_{jk}} = \frac{1}{n}\sum_{k=1}^{n} u_{ik}. \tag{14}$$

The clustering algorithm can be written as follows:

```
Algorithm: IRLS-FCM
  Step 1:  Initialize $u_{ik}, i = 1, ..., c, k = 1, ..., n$ randomly.
  Step 2:  Calculate $\boldsymbol{v}_i, i = 1, ..., c$ by using (10).
  Step 3:  Calculate $A_i, i = 1, ..., c$ by using (9).
```

Step 4: Calculate u_{ik} and $\pi_i, i = 1, ..., c, k = 1, ..., n$ by using (13) and (14).

Step 5: If iteration number exceeds the predetermined value then terminate, else go to *Step 2*.

The termination criterion is simplified in the above algorithm, but it should be recommended to check the convergence of the objective function (7).

The clustering is done on a per class basis and is implemented by using the data from one class at a time. After completing the clustering for all classes, the classification is performed by computing class memberships. Let α_q denote the mixing proportion of class q, i.e., the $a\ priori$ probability of class q. Class membership of k-th data $\boldsymbol{x}_k$ to class q is computed as:

$$u^*_{qjk} = \pi_{qj}|A_{qj}|^{-1/\gamma}/(\eta + d^2_{qjk}/0.1)^{1/\lambda}, \qquad (15)$$

$$\tilde{u}_{qk} = \alpha_q \sum_{j=1}^{c} u^*_{qjk} / \sum_{s=1}^{Q} \alpha_s \sum_{j=1}^{c} u^*_{sjk}, \qquad (16)$$

where c denotes the number of clusters of each class. The denominator in (16) can be disregarded when applied solely for classification. The FCM classifier performs somewhat better than alternative approaches do [4] and requires comparable computation time with Gaussian mixture classifier beacuse the mathematical structure is quite similar.

The modification of covariance matrices in the mixture of probabilistic principal component analysis (MPCA) [24] or the character recognition [25] is applied in our FCM classifier. P_i is an $r \times r$ matrix of eigenvectors of A_i. r equals the dimensionality of input samples. Let A'_i denotes an approximation of A_i in (9). P^p_i is an $r \times p$ matrix of eigenvectors corresponding to the p largest eigenvalues, where $p < r - 1$. Δ^p_i is a $p \times p$ diagonal matrix. p is chosen so that all A'_is are nonsingular and the classifier maximizes its generalization capability.

$$\sigma_i = (\text{trace}(A_i) - \Sigma^p_{l=1}\delta_{il})/(r - p). \qquad (17)$$

Inverse of A'_i becomes

$$A'^{-1}_i = P^p_i((\Delta^p_i)^{-1} - \sigma^{-1}_i I_p)P^{p\top}_i + \sigma^{-1}_i I_r. \qquad (18)$$

When p=0, A_i is reduced to a unit matrix and d_{ik} in (8) is reduced to Euclidean distance. Then, u_{ik} in (13) is reduced to (5) when $\pi_i = 1$ for all i and one is subtracted from λ.

Parameters λ, γ, and η are optimized by the golden section search method.

III. Parameter Optimization with CV Protocol and Deterministic FCM

High performance classifiers usually have parameters to be selected. For example, SVM has the regularization parameter and kernel parameters. After selecting the best parameters by some procedure, we use the parameters to train the whole training set, and then test new unseen data. Therefore, if the performance of classifier is dependent of random initialization, we need to select parameters with the best average performance and the result of final single run on the whole training set does not necessarily guarantee the averaged accuracy. This is a crucial problem and for making our FCM classifier deterministic, we propose a way of determining initial centroids based on principal component (PC) basis vectors.

As we will show in the numerical experiment section, the proposed classifier with two clusters for each class (i.e., c=2) performs well, so we let c=2. $\boldsymbol{p}^*_1$ is a PC basis vector of data set $D = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)^\top$ of a class, which is associated with the largest singular value σ^*_1.

Initial locations of the two cluster centroids for the class are given by

$$\begin{aligned} \boldsymbol{v}_1 &= \boldsymbol{v}^* + \sigma^*_1\boldsymbol{p}^*_1, \\ \boldsymbol{v}_2 &= \boldsymbol{v}^* - \sigma^*_1\boldsymbol{p}^*_1, \end{aligned} \qquad (19)$$

where $\boldsymbol{v}^*$ is the class mean vector. We choose the initial centroids in this way, since we know that, for a normal distribution $\mathcal{N}(\mu, \sigma^2)$, the probability of encountering a point outside $\mu \pm 2\sigma$ is 5% and outside $\mu \pm 3\sigma$ is 0.3%.

When many cluster centroids are necessary, we set as:

$$\begin{aligned} \boldsymbol{v}_1 &= \boldsymbol{v}^* + \sigma^*_1\boldsymbol{p}^*_1 + \sigma^*_2\boldsymbol{p}^*_2, \\ \boldsymbol{v}_2 &= \boldsymbol{v}^* - \sigma^*_1\boldsymbol{p}^*_1 + \sigma^*_2\boldsymbol{p}^*_2, \\ \boldsymbol{v}_3 &= \boldsymbol{v}^* + \sigma^*_1\boldsymbol{p}^*_1 - \sigma^*_2\boldsymbol{p}^*_2, \\ \boldsymbol{v}_4 &= \boldsymbol{v}^* - \sigma^*_1\boldsymbol{p}^*_1 - \sigma^*_2\boldsymbol{p}^*_2. \end{aligned} \qquad (20)$$

or it can be written more generally as:

$$\begin{aligned} \boldsymbol{v}_i = \boldsymbol{v}^* \pm \sigma^*_1\boldsymbol{p}^*_1 \pm \sigma^*_2\boldsymbol{p}^*_2 \pm ... \pm \sigma^*_m\boldsymbol{p}^*_m, \\ (i = 1, ..., 2^m, 2 \leq m \leq r). \end{aligned} \qquad (21)$$

FCM classifier has some parameters, whose best values are not known beforehand, consequently some kind of model selection (parameter search) must be done. The goal is to identify good values so that the classifier can accurately predict unseen data (i.e., testing/checking data). Because it may not be useful to achieve high training accuracy (i.e., classifiers accurately predict training data whose class labels are known), a common way is to separate training data to two parts of which one is considered unknown in training the classifier. Then the prediction accuracy on this set can more precisely reflect the performance on classifying unknown data. The cross-validation procedure can prevent the overfitting problem. In 10-fold cross-validation (10-CV), we first divide the training set into 10 subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining 9 subsets. Thus, each instance of the whole training set is predicted once so the cross-validation error rate is the percentage of data which are misclassified. The best setting of the parameters is picked via 10-CV and a recommend procedure is "grid-search". The grid-search is a methodologically simple algorithm and can be easily parallelized while many of advanced methods are iterative processes, e.g. walking along a path, which might be difficult for parallelization. In our proposed approach, the grid-search is applied for λ in the unsupervised clustering. We denote this value as λ^*.

The golden section search [26] is a technique with iterative processes for finding the extremum (minimum or maximum) of a mathematical function, by successively narrowing brackets by upper bounds and lower bounds. The technique derives its name from the fact that the most efficient bracket ratios are in a golden ratio. In our proposed approach parameters λ, η and γ are optimized in the post-supervised classification phase by using the golden section search method applied to the parameters one after another. The parameters are initialized randomly and updated iteratively, and this procedure is repeated many times, so the procedure can be parallelized.

Our parameter optimization (POPT) algorithm by grid search and golden section search is as follows:

Algorithm: POPT

```
Step 1: Initialize $\boldsymbol{v}_i, i = 1, 2$ by using (21) and set lower limit ($LL$) and upper limit ($UL$) of $\lambda^*$. Let $\lambda^* = LL$.
Step 2: Partition the training set in 10-CV by IRLS-FCM clustering with $\gamma = \eta = 1$. The clustering is done on a per class basis, then all $A_i$'s and $\boldsymbol{v}_i$'s are fixed. Set t:=1.
Step 3: Choose $\gamma$ and $\eta$ randomly from interval [0.1 50].
Step 4: Optimize $\lambda$ for the test set in 10-CV by the golden section search in interval [0.1 1].
Step 5: Optimize $\gamma$ for the test set in 10-CV by the golden section search in interval [0.1 50].
Step 6: Optimize $\eta$ for the test set in 10-CV by the golden section search in interval [0.1 50].
Step 7: If iteration $t < 50$, $t := t + 1$, go to Step 3 else go to Step 8.
Step 8: $\lambda := \lambda + 0.1$. If $\lambda > UL$, terminate else go to Step 2.
```

In the grid search for λ^* and golden section search for λ, γ and η, the best setting of the parameters is picked via 10-CV, which minimizes the error rate on test sets. The iteration number for clustering is fixed to 50, which is adequate for the objective function to converge in our experiments. More small iteration number such as 20 may be enough for some specific data sets.

When p=0, A_i is reduced to a unit matrix and d_{ik} in (8) is reduced to Euclidean distance. So we change only λ by the golden section search method and set $\pi_{qj} = 1, \alpha_q = 1$ for all j and q.

IV. Imputation of Missing Values

In this section, we propose an approach to clustering and classification without eliminating or ignoring missing values but with estimating the values. Since we are not concerned about probability distribution such as multivariate normal, instead of using the terminology "conditional expectation", the estimation is done by the least square method of Mahalanobis distances (8). The first term of (7) is the weighted sum of Mahalanobis distances between data points and cluster centroids. The missing values are some elements of data vector $\boldsymbol{x}_k$, which are estimated by the least square technique, that is, the missing elements are the solution to the system of linear equations derived from differentiating (8) with respect to the missing elements of data vector $\boldsymbol{x}_k$.

Let $x^i_{kl}, l = 1, ..., r$, be the elements of centered data $\boldsymbol{x}^i_k$ (i.e., $x^i_{kl} = x_{kl} - v_{il}$) and the j-th element x^i_{kj} be a missing element. The objective function for minimizing Mhalanobis distance with respect to the missing value can be written as:

$$L = \boldsymbol{x}^{*\top} A_i^{-1} \boldsymbol{x}^* - \boldsymbol{\mu}(\boldsymbol{x}^* - \boldsymbol{x}^i_k), \tag{22}$$

where $\boldsymbol{x}^*$ is the vector of decision variables and $\boldsymbol{\mu}$ is the vector of Lagrange multipliers. The elements of $\boldsymbol{\mu}$, $\boldsymbol{x}^*$ and $\boldsymbol{x}^i_k$ corresponding to the missing values are zero.

Then, the system of linear equations can be written as:

$$\begin{pmatrix} 2A_i^{-1} & U \\ U & Z \end{pmatrix} \boldsymbol{x}^* = \boldsymbol{b}^i_k, \tag{23}$$

where

$$U = \mathrm{diag}(1 \cdots 1 \ 0_j \ 1 \cdots 1), \tag{24}$$

$$\boldsymbol{x}^* = (x^*_1 \cdots x^*_r \ \mu_1 \cdots \mu_{j-1} \ 0 \ \mu_{j+1} \cdots \mu_r), \tag{25}$$

and

$$\boldsymbol{b}^i_k = (0_1 \cdots 0_r \ x^i_{k1} \cdots x^i_{k\ j-1} \ 0 \ x^i_{k\ j+1} \cdots x^i_{kr}). \tag{26}$$

"diag" denotes diagonal matrix and 0_j denotes that the j-th element is zero. Z is an $r \times r$ zero matrix. When more than two elements of $\boldsymbol{x}_k$ are missing, corresponding elements in (24)-(26) are also replaced by 0. All zero rows and all zero columns are eliminated from (23) and then we obtain the least square estimates of all the missing values by adding the element of centroid v_{ij} to x^*_j. We use this estimation method both for clustering and classification. Note that the clustering is done on a per class basis.

Figs.1-2 show the classification and imputation results on the well known Iris plant data by the FCM classifier with $\lambda = 0.5, \gamma = 2.5, \eta = 1$. Only the two variables, namely x_3 and x_4 are used and the problem is binary classification. These figures exemplifies the case where we can easily undestand that two clusters are suitable for a class of data with two separate distributions. Open and closed marks of triangle and circle represent the classification decision and the true class (ground truth) respectively. The five artificial data with a missing feature value are added and the values are depicted by solid line segments. Squares mark the estimated values. A vertical line segment shows the x_3 coordinate of an observation for which the x_4 coordinate is missing. A horizontal line segment shows the x_4 coordinate for which the x_3 coordinate is missing. Because the number of clusters is one ($c = 1$) for each class in Fig.1, the single class consisted of two subspecies (i.e., Iris setosa and Iris verginica) forms a slim and long ellipsoidal cluster. Therefore, the missing values on the right side and on the upper side are estimated at upper right corner of the figure. This problem is alleviated when the number of cluster is increased by one for each class as shown in Fig.2.

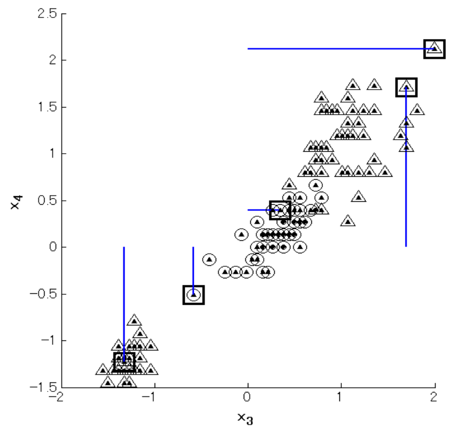

Fig. 1. Classification (triangle and circle marks) and imputation (square marks) on Iris 2-class 2D data with missing values by FCM classifier with single cluster for each class ($c = 1$).

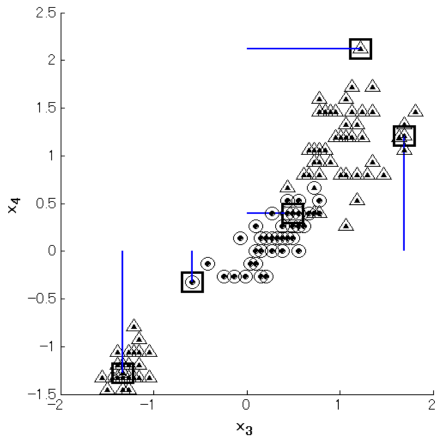

Fig. 2. Classification (triangle and circle marks) and imputation (square marks) on Iris 2-class 2D data with missing values by FCM classifier with two clusters for each class ($c = 2$).

V. Numerical Experiments

We used 8 data sets of Iris plant, Wisconsin breast cancer, Ionosphere, Glass, Liver disorder, Pima Indian diabetes, Sonar and Wine as shown in Table I. These data sets are available from the UCI ML repository (http://www.ics.uci.edu/~ mlearn/) and were used for comparisons among several prototype-based methods in [10]. Incomplete samples in the breast cancer data set were eliminated from the training and test sets. All categorical attributes were encoded with multivalue (integer) variables. And, then all attribute values were normalized to zero mean and unit variance.

Iris is the set with three classes, though it is known that Iris setosa is clearly separated from the other two subspecies [11] and Iris versicolor is between Iris setosa and Iris verginica in the feature space as shown in Fig. 1. If the problem is defined as binary one we can easily find that two clustes are necessary for the setosa-verginica class. Iris-Vc and Iris-Vg in the tables represent the Iris subspecies and each of them is two-class binary problem, i.e., one class consists of one subspecies and the other consists of remaining two subspecies. In the same way, Wine-1, Wine-2 and Wine-3 are the binary problems.

The algorithm is evaluated using 10-CV, which is applied to each data set once for the deterministic classifiers and 10 times for the classifiers with random initialization. We use a default partition into ten subsets of the same size. The performance is the average and standard deviation (s.d.) of classification errors (%).

Table II shows the results. FCM classifier based on IRLS-FCM is abbreviated to FCMC and "FCMC" column shows the results by the classifier. POPT algorithm by grid search and golden section search in section III is used. Optimum number (p) of eigen vectors is chosen from 0 (Euclidean distance) up to data dimension. Number of clusters is fixed to two, except for Sonar data where $p = 0$ and Euclidean distance is used. The iteration number for the FCM clasifier was fixed to 50, which was adequate for the objective function value to converge in our experiments.

For comparison with SVM, we used downloadable SVM toolbox for MATLAB interface to SVM^{light}[27] by Anton Schwaighofer (http://ida.first.fraunhofer.de/~ anton/software.html). The decomposition algorithm is implemented for the training routine, together with efficient working set selection strategies, which is based on random initialization. Average error rate and standard deviation by 10 separate runs of 10-CV are shown. The parameter of the RBF kernel is optimized for each run by the grid search method, so the different parameter values are used for each run. Since SVM is basically for binary classification, we used the binary problems of Iris and Wine. The classification software DTREG (http://www.dtreg.com/index.htm) has SVM option. The benchmark test results (10-CV) placed on the DTREG web site reports the error rate for the multi-class cases, i.e., 3% on Iris, 34% on Glass, and 1% on Wine.

The third column of the upper table shows the best result among six variants of C4.5 using 10 complete runs of 10-CV reported in [29]. The similar best average error rate among C4.5, Bagging C4.5 and Boosting C4.5 repoted in [30] is also displayed in parentheses (standard deviation is not reported).

Nearest neighbor classifier does not abstract the data, but rather uses all training data to label unseen data objects with the same label as the nearest object in the training set. The nearest neighbor classifer easily overfits to the training data. Accordingly, instead of 1-nearest neighbor, generally k nearest neighboring data objects are considered in k-NN classifier. Then, the class label of unseen objects is established by majority vote. For the parameter of k-NN (i.e.,k), we tested all integer values from 1 to 50. LVQ algorithm we used is LVQ1, which was a top performer (averaged over several data sets) in [28]. Initial value of the learning constant of LVQ is set as 0.3 and is changed as in [10], [28], i.e., $\beta(t+1) = \beta(t) \times 0.8$ where t (=1, ..., 100) denotes iteration number. For the parameter of LVQ (i.e., c), we tested all integer values from 1 to 50.

For GMC, the number of clustes c was chosen from 1 or 2, and optimum number (p) of eigen vectors is chosen similarly with FCMC. GMC frequently suffers from the problem of singular matrices and we need to decrease the number of eigenvectors (p) for approximating covariance matrices, though the FCM classifier alleviates the problem.

We show in Table IV the results on the benchmark datasets by artificially deleting values. These results were obtained by deleting, at random, observations from a proportion of the instances. The rate of missing feature values with respect to the whole dataset is 25%. From Iris data for example, 150 feature values are randomly deleted. The classification process by 10-CV with a default partition was repeated 10 times for the classifiers with random initializations.

In Table IV, FCMC uses missing value imputation method by the least square Mahalanobis distances. The classification error rates decay only slightly, though the proportion of instances with missing values is large.

FCMC(M) stands for the FCM classifier with the mean imputation method. Global mean is zero since the data are stnadardized to zero mean and zero is substituted for the missing value. The proposed FCMC is better than the zero imputation method.

GMC uses EM algorithm and conditional expectation for missing value imputation.

k-NN(NN) is the k-NN classifier with the nearest neighbor imputation method [15]. When the dataset volume is not extremely large, the NN imputation is an efficient method for dealing with missing values in supervised classification. The NN imputation algorithm is as follows:

Step 1: Divide the data set D into two parts. Let D_m be the set containing the instances in which at least one of the features is missing. The remaining instances with complete feature information form a set called D_c.

Step 2: Divide the instance vector into observed and missing parts as $\boldsymbol{x} = [\boldsymbol{x}_o; \boldsymbol{x}_m]$.

Step 3: Calculate the distance between the $\boldsymbol{x}_o$ and all the instance vectors from D_c. Use only those features in the instance vectors from the complete set D_c, which are observed in the vector $\boldsymbol{x}$.

Step 4: Impute missing values from the closest instance vector (nearest neighbor).

Note that all training instances must be stored in computer memory for NN imputation. Sufficient amount of complete data is needed, otherwise it may happen that no complete data exists for substituting the missing value and the computation unexpectedly terminates. k-NN classifier unexpectedly terminated for Ionosphere and Sonar data due to the lack of complete data for nearest neighbor imputation and the result is denoted by "-" in the k-NN(NN) column of Table IV.

As shown in Table III and V, different parameter values are chosen depending on the data sets without or with missing values. When λ^* (a.k.a. fuzzifier) is large, cluster centers come closer each other, so the λ^* values determin the position of centroids. We see from Tables III and V, λ assumes different values from λ^*, which optimizes classifier performance in the post-supervise phase.

Different types of models work best for different types of data, though the FCM classifier outperforms well established classifiers for almost all data sets used in our experiments.

TABLE I

DATA SETS USED IN THE EXPERIMENTS

	features	objects	classes
Iris	4	150	3
Breast	9	683	2
Ionosphere	33	351	2
Glass	9	214	6
Liver	6	345	2
Pima	8	768	2
Sonar	60	208	2
Wine	13	178	3

TABLE II

CLASSIFICATION ERROR RATES BY 10-FOLD CV WITH A DEFAULT PARTITION.

	FCMC	SVM	C4.5
Iris	**1.33**	–	5.7 ± 1.3 (4.80)
Iris-Vc	**0.67**	3.33 ± 0.63	–
Iris-Vg	**1.33**	2.73 ± 0.49	–
Breast	**2.79**	2.69 ± 0.17	5.1 ± 0.4 (4.09)
Ionosphere	**3.43**	4.60 ± 0.31	–
Glass	**28.10**	–	27.3 ± 1.5 (23.55)
Liver	**27.94**	29.65 ± 0.76	33.2 ± 1.4
Pima	**22.50**	22.38 ± 0.44	25.0 ± 1.0 (23.63)
Sonar	**10.50**	14.75 ± 1.36	24.6 ± 2.7 (19.62)
Wine	**0.00**	–	5.6 ± 1.0
Wine-1	**0.00**	0.47 ± 0.25	–
Wine-2	**0.00**	1.12 ± 0.33	–
Wine-3	**0.00**	0.59 ± 0.00	–
	k-NN	LVQ1	GMC
Iris	2.67	5.40 ± 0.87	2.00 c=1
Iris-Vc	2.67	4.87 ± 0.79	2.80 ± 0.98 c=2
Iris-Vg	2.67	4.40 ± 0.80	4.00 c=1
Breast	2.65	3.16 ± 0.16	2.97 ± 0.13 c=2
Ionosphere	13.43	10.60 ± 0.35	5.71 c=1
Glass	27.62	30.10 ± 1.38	42.38 c=1
Liver	32.65	35.24 ± 1.44	31.68 ± 1.01 c=2
Pima	23.42	24.11 ± 0.52	25.13 c=1
Sonar	13.50	14.85 ± 2.07	17.35 ± 2.19 c=2
Wine	1.76	2.35 ± 0.00	0.59 c=1
Wine-1	1.18	1.59 ± 0.46	0.00 c=1
Wine-2	1.76	3.00 ± 0.49	1.18 c=1
Wine-3	0.59	1.35 ± 0.27	0.00 c=1

VI. CONCLUSION

We have proposed an approach to classifier design within the frame work of FCM clustering.

1) The optimized FCM classifier with deterministic initialization of cluster centroids surpassed well established classifiers such as SVM, k-NN, and C4.5.

2) The accuracy of the FCM classifier with missing value imputation option only slightly deteriorates according to the increase of missing values.

TABLE III
OPTIMIZED PARAMETER VALUES USED FOR FCM CLASSIFIER WITH TWO CLUSTERS FOR EACH CLASS ($c = 2$).

	λ^*	λ	γ	η	p	
Iris	0.5	1.0000	8.8014	30.7428	3	
Iris-Vc	0.3	0.1118	0.6177	31.3914	4	
Iris-Vg	0.1	0.8185	0.6177	31.3914	2	
Breast	0.2	0.1000	–	1.0000	0	
Ionosphere	0.6	0.4866	21.7859	49.3433	4	
Glass	1.6	0.1000	2.6307	18.0977	4	
Liver	0.1	0.4703	7.7867	30.9382	4	
Pima	0.7	0.1812	2.0955	17.1634	1	
Sonar	0.2	0.3125	–	1.0000	0	$c = 16$
Wine	0.4	0.9927	16.3793	10.4402	4	
Wine-1	0.1	0.5176	7.3816	10.4402	4	
Wine-2	0.4	0.1000	4.2285	25.5607	4	
Wine-3	0.2	0.1812	4.6000	4.5229	4	

TABLE IV
CLASSIFICATION ERROR RATES (%) ON BENCHMARK DATSETS BY ARTIFICIALLY DELETING VALUES (25%). THE RESULTS OF 10-CV WITH A DEFAULT PARTITION.

	FCMC	FCMC(M)	GMC	k-NN(NN)
Iris	**2.67**	4.67	3.33 c=1	9.33
Iris-Vc	**2.67**	5.33	4.67 c=1	9.33
Iris-Vg	**2.67**	6.00	4.67 c=1	6.00
Breast	**3.68**	3.97	3.93 $\pm$ 0.11 c=2	4.12
Ionosphere	**4.86**	5.43	7.26 $\pm$ 0.68 c=2	–
Glass	**34.76**	35.71	44.29 c=1	48.10
Liver	**33.82**	34.71	39.12 c=1	37.06
Pima	**25.53**	25.00	28.16 c=1	26.05
Sonar	**13.50**	14.00	19.50 c=1	–
Wine	**1.76**	3.53	2.94 c=1	10.59
Wine-1	**0.59**	1.76	3.35 $\pm$ 0.95 c=2	4.71
Wine-2	**2.94**	3.53	3.53 c=1	8.82
Wine-3	**0.59**	2.35	1.76 c=1	3.53

3) Computation for classifying unseen instances and imputing missing values is not intense since the FCM classifier does not require to store all training data in computer storage, hence the FCM classifier outferforms k-NN classifier and NN imputation method in computational efficency.

For parameter optimization, we adopted the simple grid and golden section search, which are by no means the most efficient methods and more efficient ways are under investigation.

TABLE V
OPTIMIZED PARAMETER VALUES USED FOR BENCHMARK DATSETS BY ARTIFICIALLY DELETING VALUES (25%). TWO CLUSTERS FOR EACH CLASS ($c = 2$) ARE USED.

	λ^*	λ	γ	η	p	
Iris	0.8	0.1000	22.4641	10.4402	3	
Iris-Vc	0.4	0.4438	9.5365	13.6211	4	
Iris-Vg	0.6	0.1000	4.2285	25.5607	3	
Breast	0.3	0.1000	–	1.0000	0	
Ionosphere	0.7	0.3390	14.2559	48.9374	4	
Glass	1.9	0.5751	17.2877	37.1569	4	
Liver	0.5	1.0000	4.2285	25.5607	4	
Pima	0.7	0.4319	33.7207	26.3013	1	
Sonar	0.2	0.2313	–	1.0000	0	$c = 16$
Wine	0.6	0.4821	12.2867	26.3013	4	
Wine-1	0.6	0.3863	4.6000	10.4402	4	
Wine-2	0.5	0.1000	2.0535	22.3347	4	
Wine-3	0.1	0.9927	8.0370	13.6211	4	

REFERENCES

[1] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, 1981.
[2] P. W. Holland and R. E. Welsch, “Robust regression using iteratively reweighted least-squares, ”*Communications in Statistics*, vol. A6, no. 9, pp. 813-827, 1977.
[3] H. Ichihashi and K. Honda, “Fuzzy c-means classifier for incomplete data sets with outliers and missing values,” *Proc. of the International Conference on Computational Intelligence for Modelling, Control and Automation*, Vienna, November, pp.457-564, 2005..
[4] H. Ichihashi, K. Honda, T. Hattori, “Regularized discriminant in the setting of fuzzy c-means classifier,” *Proc. of the IEEE World Congress on Computational Intelligence*, Vancouver, Canada, pp.4266-4271, 2006.
[5] H. Ichihashi, K. Honda, F. Matsuura, “ROC analysis of FCM classifier with Cauchy weight,” *Proc. of the 3rd International Conference on Soft Computing and Intelligent Systems*, Tokyo, Japan, pp.1912-1917, 2006.
[6] V.N. Vapnik, *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, Berlin, 1982.
[7] C. Cortes and V. Vapnik, “Support-vector network.” *Machine Learning*, vol.20, pp.273-297, 1995.
[8] J. R. Quinlan, *C4.5, Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993.
[9] T. Kohonen, *Self-Organization and Associative Memory*, 3rd Edn. Springer, Berlin, 1989.
[10] C.J. Veenman and M.J.T. Reinders, “The nearest sub-class classifier: a compromise between the nearest mean and nearest neighbor classifier,” *IEEE Transactions on PAMI*, vol.27, no.9, pp.1417-1429, 2005.
[11] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
[12] R. L. Streit and T. E. Luginbuhl, “Maximum likelihood training of probabilistic neural networks,” *IEEE Transactions on Neural Networks*, vol.5, no.5, pp.764-783, 1994.
[13] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
[14] O. Troyanskaya, *et.al.*, “Missing value estimation methods for dna microarrays,” *Bioinformatics*, vol.17, no.6, pp.520-525, 2001.
[15] J. K. Dixon, “Pattern recognition with partly missing data,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol.SMC-9, no.10, pp.617-621, 1979.
[16] R. J. Little and D.B. Rubin, *Statistical Analysis with Missing Data, Second Edition*, John Wiley and Sons, New York, 2002.
[17] K. Honda, H. Ichihashi, “Linear fuzzy clustering techniques with missing values and their application to local principal component analysis,” *IEEE Trans. Fuzzy Syst.*, vol.12, no.2, pp.183-193, 2004.
[18] K. Honda, H. Ichihashi, “Component-wise robust linear fuzzy clustering for collaborative filtering,” *Int. J. Approximate Reasoning.*, vol.37, no.2, pp.127-144, 2004.
[19] S. Miyamoto, D. Suizu, O. Takata, “Methods of fuzzy c-means and possibilistic clustering using a quadratic term, *Scientiae Mathematicae Japonicae* vol.60, no.2, pp.217-233, 2004.
[20] H. Ichihashi, K. Honda, A. Notsu and T. Hattori, “Aggregation of Standard and Entropy Based Fuzzy c-Means Clustering by a Modified Objective Function,” *Proc. of the 2007 IEEE Symposium on Foundations of Computational Intelligence*, Hawaii, April, 2007 (to appear).
[21] D. E. Gustafson and W. C. Kessel, “Fuzzy clustering with a fuzzy covariance matrix, ”*Proc. IEEE CDC*, vol.2, pp.761-766, 1979.
[22] T. A. Runkler and J. C. Bezdek, “Alternating cluster estimation: a new tool for clustering and function approximation,” *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 4, pp. 377-393, 1999.
[23] P. J. Huber. *Robust Statistics*. New York:Wiley, first edition, 1981.
[24] M.E. Tipping and C.M. Bishop, “Mixtures of probabilistic principal component analysers, ”*Neural Computation*, vol.11, pp.443-482, 1999.
[25] F. Sun, S.Omachi, and H. Aso, “Precise selection of candidates for hand written character recognition,” *IEICE Trans. Information and Systems*, vol.E79-D, no.3, pp.510-515, 1996
[26] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C, The Art of Scientific Computing, second edition*, Cambridge University Press, Cambridge, 1999.
[27] T. Joachims, “Making large-scale SVM learning practical,” *In: B.*

Scholkopf et al.(ed.), Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, pp.169-184, 1999.

[28] J. C. Bezdek and L. I. Kuncheva, "Nearest prototype classifier designs: An experimental study," *Int. J. of Intelligent Systems*, vol.16, pp.1445-1473, 2001.

[29] T. Elomaa and J. Rousu, "General and efficient multisplitting of numerical attributes,"*Machine Learning*, vol. 36, no. 3, pp. 201-244, 1999.

[30] J. R. Quinlan, "Bagging, Boosting, and C4.5,"*Proc. 13th National Conference on Artificial Intelligence, AAAI/IAAI*, vol. 1, pp. 725-730, Portland, Oregon, August, 1996.