Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2009, Article ID 360834, 9 pages
doi:10.1155/2009/360834

Research Article

Incremental Local Linear Fuzzy Classifier in Fisher Space

Armin Eftekhari,[1] Hamid Abrishami Moghaddam,[1, 2] Mohamad Forouzanfar,[3] and Javad Alirezaie[1, 4]

[1] *Faculty of Electrical Engineering, K.N. Toosi University of Technology, P. O. Box 16315-1355, Tehran, Iran*
[2] *Unité de Génie Biophysique et Médical, Groupe de Recherche sur l'Analyse Multimodale de la Fonction Cérébrale (GRAMFC), Faculté de Médecine, 3 rue des Louvels, 80036 AMIENS cedex, France*
[3] *School of Information Technology and Engineering, University of Ottawa, 800 King Edward Avenue, Ottawa, ON, Canada K1N 6N5*
[4] *Department of Electrical and Computer Engineering, Ryerson University, 350 Victoria Street, Toronto, ON, Canada M5B 2K3*

Correspondence should be addressed to Hamid Abrishami Moghaddam, moghadam@eetd.kntu.ac.ir

Received 4 August 2008; Revised 4 March 2009; Accepted 25 March 2009

Recommended by Christophoros Nikou

Optimizing the antecedent part of neurofuzzy system is an active research topic, for which different approaches have been developed. However, current approaches typically suffer from high computational complexity or lack of ability to extract knowledge from a given set of training data. In this paper, we introduce a novel incremental training algorithm for the class of neurofuzzy systems that are structured based on local linear classifiers. Linear discriminant analysis is utilized to transform the data into a space in which linear discriminancy of training samples is maximized. The neurofuzzy classifier is then built in the transformed space, starting from the simplest form (a global linear classifier). If the overall performance of the classifier was not satisfactory, it would be iteratively refined by incorporating additional local classifiers. In addition, rule consequent parameters are optimized using a local least square approach. Our refinement strategy is motivated by LOLIMOT, which is a greedy partition algorithm for structure training and has been successfully applied in a number of identification problems. The proposed classifier is compared to several benchmark classifiers on a number of well-known datasets. The results prove the efficacy of the proposed classifier in achieving high performance while incurring low computational effort.

1. Introduction

Both fuzzy logic and neural networks include approaches to human-like reasoning that utilize the human tolerance for incompleteness, uncertainty, imprecision, and fuzziness in a decision making process. Fuzzy logic is a key tool to express the knowledge of domain experts so that valuable experience of humans can be incorporated into the system design. The neural network is an information processing system with the ability to learn from training data. The learning capability of neural networks makes them an appropriate choice for combination with fuzzy systems in order to automate or support the process of developing a fuzzy system for a given task [1]. In this view, neurofuzzy systems have been introduced and widely investigated [2]. A neurofuzzy system is a fuzzy system that is trained by a learning algorithm derived from neural network theory. The learning procedure is performed by interleaving the optimization of the antecedent and consequent part parameters. The performance of a neurofuzzy system is largely influenced by structure learning which involves two major issues: (i) parameter tuning of the antecedent part, which provides us with the fuzzy partitioning of the input space. (ii) Parameter tuning of consequent part in which the parameters of consequent functions are obtained. Each subspace together with its associated consequent function is used to characterize a corresponding fuzzy rule [3]. Generally, the local models (consequent functions) are chosen to be linear, which yields local linear model structures [4].

Recently, neurofuzzy systems have found extensive applications in pattern recognition [5–7]. In this context, several techniques for deriving fuzzy rules from training data such as fuzzy clustering and partitioning-based methods have been proposed. The fuzzy clustering-based methods search the input space for clusters, which are then projected to each dimension of input space to gain fuzzy rules with better interpretability. This approach encompasses a variety of algorithms such as Kohonen learning rule, hyperbox method, product-space partitioning, and fuzzy C-mean method [8]. Examples of partitioning-based methods are NEFCLASS and NEFCAR, which start with a large number of partitions. These partitions are then pruned to select the best-performing fuzzy rules [1, 7, 9]. For a detailed discussion on neurofuzzy rule generation algorithms, the reader is referred to [10–14].

This study proposes a novel incremental technique for structure optimization of local linear neurofuzzy classifiers. The proposed neurofuzzy classifier is built starting from the most generic and simplest form (a global linear classifier). If the overall performance of the classifier was not satisfactory, it would be iteratively refined by incorporating additional local classifiers. Proposed refinement strategy is motivated by LOLIMOT, which is a greedy partition algorithm for structure training of local linear neurofuzzy models that determine the (sub) optimal partitioning of input space by axis-orthogonal splits [15, 16] and has found extensive applications in identification problems due to fast implementation and high accuracy. Adoption of LOLIMOT algorithm to classification requires inevitable modifications. Conventional LOLIMOT is restricted to axis-orthogonal splits and is unable of handling high-dimensional data. We address these problems by employing a well-known statistical stage, namely, linear discriminant analysis (LDA). Therefore, antecedent structure of neurofuzzy classifier is built in the transformed (and if needed reduced) input space by axis-orthogonal splits. Moreover, for proper adoption of LOLIMOT algorithm to classification, a novel interpretation of error is introduced. Once the antecedent parameters are determined, rule consequent parameters are efficiently estimated using a local least square approach. To assess the performance of the proposed method, results are compared with conventional classifiers (neural networks, linear Bayes, and quadratic Bayes), neurofuzzy classifiers (NEFCLASS and FuNe I), piecewise linear classifiers, and decision trees (C4.5). Experimental results on several well-known datasets demonstrate that, in most cases, our algorithm outperforms state-of-the-art classifiers and significantly improves the classification results. The rest of this paper is organized as follows. In Section 2, local linear neurofuzzy classifiers are introduced and common approaches for antecedent and consequent parameter optimization are discussed. In Section 3, our proposed classifier is developed. Section 4 is dedicated to assessment of the proposed algorithm and the paper is concluded in Section 5.

2. Local Linear Neurofuzzy Classifier

A neurofuzzy system with multiple outputs can be realized either by a single SIMO or MIMO model or by a bank of SISO or MISO models [17]. In the current study, the former approach is pursued as it often requires fewer neurons [16]. Assume a set of input/label pairs $\{U, Y\}$, where $U \in \mathbb{R}^p$ and $Y \in \{0,1\}^K$. In the case of two-class problems, it is most convenient to use the binary representation, in which there is a single target variable $y \in \{0,1\}$ such that $y = 1$ represents first class and $y = 0$ represents the other class. When facing a K-class problem, it is often convenient to use a 1-of-K coding scheme in which the label Y is a vector of length K such that if the class is C_j, then all elements of Y are zero except its jth element denoted by $(Y)_j$, which takes the value 1. The elements of label vector Y can be interpreted as posterior probabilities of corresponding classes, with the values of probability taking only the extreme values of 0 and 1. Therefore, we wish to predict discrete class labels, or more generally posterior probabilities that lie in the range $(0,1)$. This is achieved by introducing an activation function $f(\cdot)$ [18] to limit the output of the model so that it falls into $(0,1)$. The choice of activation function is usually logistic sigmoid ($K = 2$ classes) or softmax ($K \geq 2$ classes). Decision is made by assigning each test sample to the class with maximum posterior probability. The network architecture of a neurofuzzy classifier, structured based on local linear classifiers, is depicted in Figure 1, where the rule antecedent inputs $Z \in \mathbb{R}^{nz}$ and the rule consequent inputs $X \in \mathbb{R}^{nx}$ are subsets of the input samples U. Each neuron $i = 1,\ldots,M$ of the model realizes a fuzzy rule:

$$R_i = \text{IF } (Z)_1 \text{ IS } A_{(i,1)} \text{ AND}\ldots\text{AND } (Z)_{nz} \text{ IS } A_{i,nz} \quad \text{THEN } \hat{Y} = W_i^T X, \tag{1}$$

where $A_{i,j}$ is the jth fuzzy set defined on ith input and $W_i \in \mathbb{R}^{nx\times K}$. Each neuron or rule represents K local linear classifiers (LLCs) and an associated validity (weighting) function that determines the region of validity of those LLCs. For a reasonable interpretation of local classifiers it is furthermore necessary that the validity functions sum up to one for any antecedent input Z. The output of the local linear neurofuzzy classifier would be

$$\hat{Y}(U) = f\left(\sum_{i=1}^{M} \varphi_i(Z)\cdot \hat{Y}_i(X)\right) = f\left(\sum_{i=1}^{M} \varphi_i(Z)\cdot \left(W_i^{\mathrm{T}} X\right)\right), \tag{2}$$

where $\hat{Y}_i$ denotes the output of local models and $\varphi_i(\cdot)$ is interpreted as weighting function, $i = 1,\ldots,M$. Thus, the output of the model is obtained by applying $f(\cdot)$ to the weighted sum of the outputs of the LLCs. In other words, the model interpolates between local models by weighting functions. In the following, the validity functions $\varphi_i(\cdot)$ are chosen to be normalized Gaussians:

$$\varphi_i(Z) = \frac{\mu_i(Z)}{\sum_{j=1}^{M} \mu_j(Z)}$$
$$\mu_{\mathrm{i}}(Z) = \exp\left(-\frac{1}{2}(Z - C_i)^{\mathrm{T}} {\textstyle\sum_{\mathrm{i}}^{-1}} (Z - C_i)\right), \tag{3}$$

where $C_i \in \mathbb{R}^{nz}$ is the center of ith membership function and $\sum_i \in \mathbb{R}^{nz\times nz}$ is a diagonal matrix containing variances of individual dimensions, that is, $\sum_i = \mathrm{diag}\{\sigma_{i,1}^2,\ldots,\sigma_{i,nz}^2\}$.

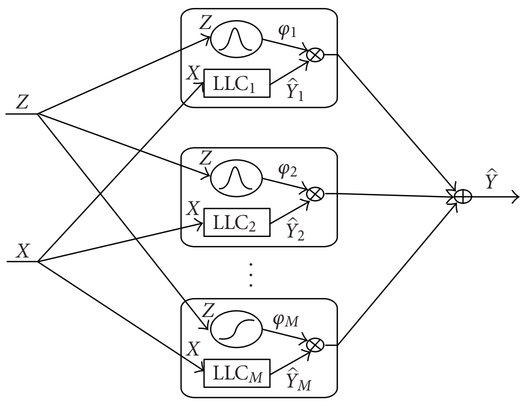

FIGURE 1: Local linear neurofuzzy classifier.

Here, we will assume the most general case for X and Z, where $X = Z = U$. It should be pointed out that in contrast to the models used for identification, discussed neurofuzzy classifier will be no longer linear in the consequent parameters due to the presence of $f(\cdot)$. This will lead to more analytical and computational complexities than for identification models. However, at the expense of losing the probabilistic point of view, we can omit the nonlinear activation function as in [19]. A test sample is then assigned to the class with maximum activation value and the classifier would be linear in consequent parameters. Optimization of rule antecedent structure and rule consequent parameters is discussed in the following sections.

2.1. Rule Consequent Parameters. Rule consequent parameters are interpreted as parameters of local classifiers. Due to linearity assumption for activation function, the neurofuzzy classifier presented by (2) is linear in consequent parameters. Therefore, these parameters can be efficiently estimated from training patterns using a least square approach, provided that the rule antecedent structure is given. Simultaneous optimization of all consequent parameters (global optimization) yields the best results in the sense of least mean square error but involves extreme computational effort. Alternatively, we can use local estimation approach presented in [15], which neglects the overlap between the validity functions and estimates the parameters of each rule separately. This approach is computationally more efficient than global estimation. The cost, however, is the introduction of a bias error while, on the other hand, the variance error (and the effect of over-fitting) is decreased and more robustness to noise is gained. In this paper, the local estimation approach is pursued and is described as follows. Instead of estimating all $M \times K \times (p+1)$ consequent parameters simultaneously (as in global estimation), M local estimations are carried out for the $K \times (p+1)$ parameters of each neuron. Note that the parameter matrix associated with ith LLC is $W_i \in \mathbb{R}^{(p+1)\times K}$ and that the contribution of ith LLC to the output vector $\hat{Y}_i(U)$ is $\varphi_i(U)(W_i^{\mathrm{T}}U)$, $i = 1,\dots,M$. The contribution of ith LLC is dominant only in the region where the associated validity function $\varphi_i(\cdot)$ is close to one (which happens near the center of $\varphi_i(\cdot)$). Training samples in this region are highly relevant for the estimation of W_i. Therefore, local estimation of W_i can be achieved by performing the following weighted least square optimization:

$$\min_{W_i} I_i \quad \text{where} \quad I_i = \sum_{j=1}^{N} \varphi_i\left(U_j\right) \cdot \left\| Y\left(U_j\right) - \hat{Y}\left(U_j\right) \right\|^2, \tag{4}$$

where U_j denotes the jth input sample, $j = 1,\dots,N$. This optimization is equivalent to fitting a linear classifier to weighted training data. Let the target matrix $\Upsilon \in \mathbb{R}^{N\times K}$, the regression matrix $X \in \mathbb{R}^{N\times(p+1)}$, and the weighting matrix $Q_i \in \mathbb{R}^{N\times N}$ be defined as follows:

$$\Upsilon = \begin{bmatrix} Y_1 & Y_2 & \cdots & Y_N \end{bmatrix}^{\mathrm{T}}, \qquad X = \begin{bmatrix} 1 & U_1^{\mathrm{T}} \\ 1 & U_2^{\mathrm{T}} \\ \vdots & \vdots \\ 1 & U_N^{\mathrm{T}} \end{bmatrix},$$

$$Q_i = \begin{bmatrix} \varphi_i(U_1) & 0 & \dots & 0 \\ 0 & \varphi_i(U_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \varphi_i(U_N) \end{bmatrix}. \tag{5}$$

Then, it can be simply verified that the optimum $W_i \in \mathbb{R}^{(p+1)\times K}$ that minimizes (4) is obtained as

$$W_i = \left(X^{\mathrm{T}} Q_i X\right)^{-1} X^{\mathrm{T}} Q_i \Upsilon \tag{6}$$

2.2. Rule Antecedent Structure. Training of the antecedent parameters is a nonlinear optimization task, which provides us with the proper partitioning of the input space. Two common strategies for antecedent structure optimization are clustering-and partitioning-based techniques. In order to embed data-driven knowledge in a neurofuzzy system, clustering methods such as Fuzzy RuleNet [20] utilize cluster vectors extracted from the input dataset to initialize the centers of fuzzy rules. A learning algorithm is then applied to fine tune the rules based on the available training data. These approaches usually search for hyperellipsoidal or hyperrectangular clusters in input space and are shown to typically produce rules which are hard to interpret [21, 22]. Partitioning-based methods such as NEFCLASS [9] divide the input space into finer regions by grid partitioning. Each partition is supposed to represent an if-then rule. These rules are then pruned using some heuristics. Finally, membership functions are defined using only best performing rules. In other words, NEFCLASS does not induce fuzzy classification rules by searching for clusters, but by modifying the fuzzy partitionings defined on each single dimension. Evidently, partitioning-based approaches are computationally expensive [23].

3. Proposed Algorithm for Structure Optimization

Tuning of the antecedent parameters of a neurofuzzy system is a nonlinear optimization task, which provides us with the proper input space partitioning. Some commonly used strategies for antecedent parameter optimization were discussed in Section 2.1. In addition, optimization of consequent parameters using local least square approach was discussed in Section 2.2. The proposed algorithm for structure optimization increases the complexity of the local linear neurofuzzy classifier during the training phase. Hence, it starts with a coarse partitioning of the input space, which is then refined by increasing the resolution of the input space partitioning. The proposed algorithm is based on divide-and-conquer principle. This principle is widely used to attack complex problems by dividing them into simpler classification tasks whose resulting local classifiers are then combined to obtain a global classifier which also generalizes well [24]. Our strategy for input space partitioning is motivated by LOLIMOT, which is a local linear neurofuzzy algorithm that uses axis-orthogonal splits to avidly partition the input space for the rule antecedent parameter and structure training. LOLIMOT has been successfully applied in a number of identification problems and gained significant attention due to simple and fast optimization of rule antecedent parameters. Computational complexity of LOLIMOT grows linearly with the number of neurons and cubically with the number of consequent parameters of each neuron. This level of computation complexity is quite favorable [16]. As will be discussed shortly, adoption of LOLIMOT algorithm to classification requires inevitable modifications.

One of the most severe restrictions of LOLIMOT is the axis-orthogonal partitioning of the input space. This restriction, while being crucial for interpretation as a fuzzy system and for the development of an extremely efficient construction algorithm, leads to the following shortcomings. (i) Improper splitting of input space, which frequently happens when optimal partitioning of input space does not align with axis-orthogonal directions. In such cases, the nonlinearity of data in the original input space does not stretch along the input space axes and hence LOLIMOT cannot efficiently determine proper input partitioning [25]. (ii) Curse of dimensionality, which often plagues fuzzy systems in real-world applications. Fuzzy methods, which are computationally manageable in low-dimensional spaces, can become completely impractical in high-dimensional spaces. Since at each iteration, LOLIMOT tries all divisions of worst LLC to decide about further refinement, curse of dimensionality will be more prohibitive. Several techniques have been proposed to address these two drawbacks. For example, Nelles developed an axis-oblique decomposition algorithm, which suffers from computational concerns [25]. In addition, using different input spaces for rule antecedent and consequent was suggested in [16], which could result in the alleviation of computational efforts. Evidently, adoption of LOLIMOT to classification confronts the above shortcomings, especially when the discriminancy of classes is small in the original axes. We suggest using a computationally cheap, easy to implement statistical stage, namely, LDA (also known as Fisher discriminant analysis), which alleviates the mentioned problems by rotating the original axes, so that the linear discriminancy of training samples along the new axes is maximized in a global sense [8]. The basic concept of LDA is to seek for the most efficient projective directions which minimize the scattering of samples in each class and maximize the distance of different classes. In addition, LDA is capable of selecting the best linear combinations of input features for classification and hence can be used for dimensionality reduction. Therefore, axis-orthogonal partitioning of the transformed input space (building the structure in the transformed space) often significantly reduces the complexity of antecedent structure, as well as computational cost. LDA is formally described as follows. Consider the sample set $\{X_{i,j}\}$, where $i = 1, \dots, I$ denotes the class to which $X_{i,j}$ belongs and $j = 1, \dots, N_i$ denotes the index of the sample $X_{i,j}$ in the corresponding class. Now, between-class scatter matrix S_B is introduced as

$$S_B = \frac{1}{N}\sum_{i=1}^{I} N_i\left(\overline{X}_i - \overline{X}\right)\left(\overline{X}_i - \overline{X}\right)^{\mathrm{T}}, \tag{7}$$

where $\overline{X}_i = (1/N_i)\sum_{j=1}^{N_i} X_{i,j}$ denotes the sample mean in class I_i and $\overline{X} = (1/N)\sum_{i=1}^{I}\sum_{j=1}^{N_i} X_{i,j}$ is the global sample mean. Similarly, within-class scatter matrix is defined as

$$S_W = \frac{1}{N}\sum_{i=1}^{I}\sum_{j=1}^{N_j}\left(X_{i,j} - \overline{X}_i\right)\left(X_{i,j} - \overline{X}_i\right)^{\mathrm{T}}. \tag{8}$$

LDA then searches such optimal subspace projections which minimize the trace of the resulting within-class scatter matrix, while maximize the trace of the between-class scatter matrix. In other words, the selected features are eigenvectors of $(S_W)^{-1}S_B$ corresponding to largest eigenvalues. Another, perhaps more popular, unsupervised alternative to LDA is principal component analysis (PCA), which minimizes the information loss upon projection to the lower dimensional space. Since PCA minimizes the reconstruction error, classification based on PCA generally achieves lower performance compared to LDA, as verified in Section 4.1. For a detailed discussion on linear dimensional reduction techniques, the interested reader is referred to [8, 26, 27].

Another practical concern for adoption of LOLIMOT to classification is the error interpretation. Training of local linear models in an LOLIMOT system is achieved by minimizing the local loss function defined in (4). This loss function is also used for comparison of local models. In this paper, a novel interpretation of error is introduced. While the loss function of (4) is used to train the LLCs, we suggest using a different error index for comparison of LLCs, which is based on percentage error rather than l_2-norm ($\|\cdot\|_2$) of the classification error. The percentage error resembles the l_1-norm of error ($\|\cdot\|_1$), which is shown to gain better classification results than l_2-norm of error [28]. Through our experiments, it was found that this interpretation of error improves the classification results.

Finally, note that the standard deviation of validity functions is selected to be proportional to extension of corresponding hyperrectangle. In the current study, this proportionality factor is fixed and assumed to be 1/3 [16]. The proposed algorithm can be summarized as follows.

(1) *Finding the most discriminative basis*: apply LDA in order to find the most discriminative basis. If needed, dimension reduction is also realized in this step by keeping only the most discriminative features in the new basis. The antecedent structure is built in this transformed space.

(2) *Start with an initial model*: use any prior knowledge to construct the validity functions in the transformed initial input space partitioning. If no input space partitioning is available a priori, then set $M = 1$ and start with a single LLC.

(3) *Compare LLCs to find the worst LLC*: use the following equation to calculate the error index l for all LLCs, in which each misclassified pattern is assigned to the LLC with largest degree of validity. Then, the LLC with maximum error index l is selected as the worst-performing, which is denoted by LLC$_b$:

$$l_i = \frac{N(S_{i,e})}{N(S_i)} \quad \text{where} \quad S_i = \left\{U \mid \varphi_i(U) > \varphi_j(U) \text{ for } j \neq i\right\}$$
$$S_{i,e} = \{U \mid U \in S_i \text{ AND } U \text{ is misclassified}\}, \tag{9}$$

where, N(ξ) denotes the number of elements of vector ξ.

(4) *Check all divisions of worst LLC*: consider the LLC$_b$ for further refinement. The hyperrectangle of this LLC is partitioned into two halves with an axis-orthogonal split. Divisions in all dimensions are considered. For each of the p divisions, the following steps are taken.

   (a) Construct the membership functions for both hyperrectangles.

   (b) Construct all validity functions.

   (c) For both newly generated LLCs, weigh the training samples with corresponding validity functions and fit a linear classifier to these weighted samples by minimizing local loss function defined in (4) (local optimization of the rule consequent parameters for both newly generated LLCs).

   (d) Calculate the percentage error of classification for the current overall model.

(5) *Find the best division*: select the best of the p alternatives checked in step 4. The validity functions constructed in step 4a and the LLCs optimized in step 4c are included in the classifier. The number of LLCs is increased by one.

(6) *Test for convergence*: if the termination criterion (e.g. convergence of performance) is not met, go to step 2.

In the next section, the efficacy of the proposed framework is experimentally studied on several datasets. In addition, to provide a better insight into the procedure, operation of the algorithm will be graphically illustrated.

4. Experiments

This section presents the classification results of the proposed method on several well-known datasets. The error rates of the proposed classifier are compared to that of a number of existing pattern classification algorithms. To this end, four datasets from ELENA project [29], namely, Iris_CR, Phoneme_CR, Satimage_CR, and Texture_CR, and two datasets from UCI machine learning repository [30], namely, Wisconsin breast cancer and Sonar are selected. ELENA project and UCI machine learning repository are resources of databases for testing and benchmarking pattern classification algorithms. Main features of these datasets are summarized in Table 1.

The CR affix in the names of datasets from ELENA project indicates that the datasets are initially preprocessed by a normalization routine to center each feature and enforce unit variance. In our experiments on these datasets, we follow a similar technique to [31]. First, each dataset is partitioned into two equal random sets: one for training and the other for test phase. Then, the roles of two halves are reversed. To achieve more accurate results, experiments are repeated 20 times and the average error rate is reported.

4.1. Role of LDA in Preprocessing. In this subsection, the role of LDA in the preprocessing phase is experimentally studied. Our first experiment is conducted on Iris_CR dataset. To be able to visualize the results, PCA [8] is first applied to reduce the number of features to two. Then, the proposed algorithm is applied to obtain the partitioning of the input space for all samples in Iris_CR dataset. Using rectangles to show the validity region of the corresponding LLC's, Figures 2(a) and 2(b) depict the obtained partitioning without and with LDA preprocessing, respectively. Therefore, this illustration provides a comparison between PCA and LDA in the preprocessing phase. It is observed that there exists a partition in Figure 2(a), for which it is impossible to find a linear classifier that correctly classifies all samples in the corresponding partition, whereas this situation does not occur in Figure 2(b), in which splitting directions are not axis-orthogonal, but are selected to maximize the linear discriminancy of the samples. This illustrative example implies that, with the same number of partitions, LDA generally provides a better partitioning of the input space compared to PCA. Figure 2(b) also provides a valuable insight into the process of input space partitioning by the proposed algorithm.

As our second experiment, the advantage of using LDA preprocessing is quantified in terms of the performance of the proposed classifier on a number of datasets, namely, Iris_CR, Satimage_CR, Texture_CR, and Phoneme_CR. Table 2 lists average classification error rates of the proposed classifier, using PCA or LDA in the preprocessing phase. In accordance with our expectation, using LDA leads to better classification performance on all datasets.

4.2. Comparison with Conventional Classifiers. Table 3 lists average classification error rates of the proposed algorithm

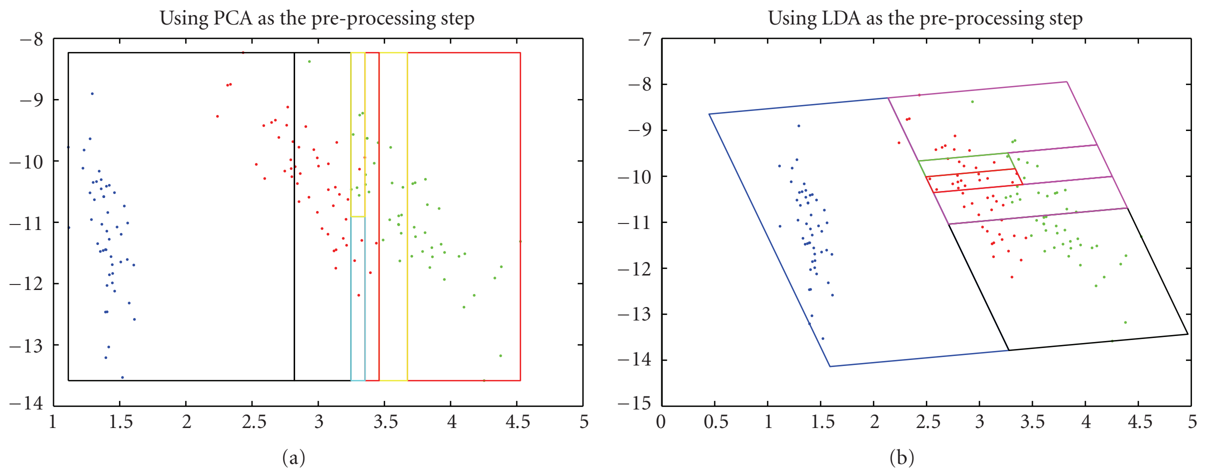

FIGURE 2: Application of the proposed algorithm for partitioning of the input space in Iris_CR dataset, (a) without and (b) with LDA in the pre-processing phase.

TABLE 1: Main features of datasets used in our experiments.

Dataset	Number of classes	Number of features	Number of patterns
Iris	3	4	150
Satimage	6	36	6435
Texture	11	40	5500
Phoneme	2	5	5404
BCW	2	9	699
Sonar	2	60	208

TABLE 2: Error rates in percentage for the proposed classifier using PCA or LDA preprocessing. The best results are highlighted in boldface.

	Datasets			
Classifiers	Iris_CR	Satimage_CR	Texture_CR	Phoneme_CR
Proposed classifier with LDA preprocessing	**2.33**	**13.54**	**2.80**	**23.15**
Proposed classifier with PCA preprocessing	4.27	77.83	19.49	27.22

TABLE 3: Error rates in percentage for conventional and proposed classifiers on several datasets. The best results are highlighted in boldface.

	Datasets			
Classifiers	Iris_CR	Satimage_CR	Texture_CR	Phoneme_CR
Neural network	4.67	16.02	5.15	**20.79**
Linear Bayes	2.67	16.69	2.58	27.00
Quadratic Bayes	4.67	14.22	**0.96**	24.59
Proposed classifier	**2.33**	**13.54**	2.80	23.15

TABLE 4: Error rates for piecewise linear classifier [32], C4.5 decision tree [33] and the proposed classifier on several datasets. The best results are highlighted in boldface.

	Datasets					
Classifiers	Iris_CR	Satimage_CR	Texture_CR	Phoneme_CR	BCW	Sonar
Piecewise linear	3.34	13.90	4.90	17.85	4.80	19.70
C4.5	7.33	16.50	11.91	**16.08**	5.26	25.60
Proposed classifier	**2.33**	**13.54**	**2.80**	23.15	**2.84**	**10.71**

and of some conventional classifiers, namely, neural network, linear Bayes, and quadratic Bayes on several datasets, as reported in [31]. It shall be noted that, in [31], each classifier has been reasonably optimized with regards to parameter settings and available features. In addition, an earnest effort was made to optimize each individual classifier with respect to selecting good values for the parameters which govern its performance. Moreover, feature selection techniques have been applied to feed each classifier with best features. Table 3 indicates that, both on Iris_CR and Satimage_CR datasets, the proposed technique outperforms other classifiers. On Texture_CR dataset, our classifier outperforms the neural network classifier, achieves results comparable to the linear Bayes classifier, and is slightly worse than the quadratic Bayes classifier. On the other hand, on Phoneme_CR dataset, our classifier outperforms both linear and quadratic Bayes classifiers and achieves results worse than the neural network classifier. These results suggest that the proposed simple local linear fuzzy classifier could be quite successful compared to these conventional classifiers. Finally, note that the proposed classifier typically achieves better results in comparison with the neural network classifier, to which it can be regarded as a close relative.

4.3. Comparison with Other Neurofuzzy Classifiers. NEFCLASS and FuNe I are two well-known neurofuzzy classifiers. NEFCLASS starts with a large number of partitions in the input space, which, as mentioned in Section 2.2, are pruned to select the best-performing fuzzy rules [9]. FuNe I, on the other hand, has a five-layer feedforward structure and restricts itself to rules with one or two antecedents. Fuzzy rules are then learned by a special training network, that helps to identify suitable combinations of one or two variables as antecedents. These rules are then trained to find suitable fuzzy sets for the rules [34, 35].

Performance of these classifiers and of the proposed method is compared on Iris_CR dataset. For NEFCLASS and FuNe I, the number of rules was limited to ten and seven, respectively. Reported error rates of NEFCLASS and FuNe I are 3.33% and 4%, respectively, while our method achieves the error rate of 2.33%, with seven partitions. It shall be pointed out that, in contrast to the proposed classifier, NEFCLASS and FuNe I suffer from high-computational complexity [35].

4.4. Comparison with Piecewise Linear Classifiers. Piecewise linear classifiers approximate the complex decision boundaries with piecewise linear functions. Recently, Kostin presented a simple and fast piecewise linear classifier which demonstrated comparable (even superior in many cases) results with many well-known benchmark classifiers [32]. Kostin's classifier is based on simple calculation of centroids of classes and the creation of a binary partition tree of class centroids, which is then used to sequentially construct piecewise linear boundaries for each nonleaf node of the partition tree [32]. As was the case in our classifier, complexity of the classifier is sequentially increased until satisfactory performance is achieved.

In this subsection, due to similar essence and properties, performance of the Kostin's classifier [32], as a representative member of piecewise linear classifiers, is compared with the proposed classifier. The average classification error rates for two methods are listed in Table 4. As indicated by the results, the proposed classifier achieves a better performance compared to Kostin's classifier on five datasets, with slightly worse performance on the Phoneme_CR dataset. This improvement is intuitively explained by noting that, with the same complexity, natural datasets are generally better expressed by space grids rather than hyperplanes. Furthermore, the fuzzy nature of the decision making process in the proposed classifier may be regarded as an advantage over crisp decision boundaries involved in [32].

4.5. Comparison with Decision Tree Classifiers. Decision tree algorithms are regarded as a powerful classification tool in machine learning society, which have appeared quite influential in practice [33]. This classifiers are constructed in a form of a decision tree, in which each nonleaf node tests a function of some attributes. An unknown pattern is then classified by making consecutive decisions starting from the root until reaching a leaf node. Clearly, proper selection of the test functions and associated attributes at each node are vital for successful application of decision tree classifiers. Among several choices, C4.5 is utilized in our experiments as a successful and popular decision tree classifier [33]. Using two-way splits for numeric attributes in the creation of the decision tree, C4.5 examines a family of possible tests at each node and selects the one which maximizes the value of some splitting criterion. Once the tree is built, a pruning procedure is performed to avoid overfitting and excessive complexity.

Due to similar essence and characteristics, comparison of the proposed classifier and decision tree classifiers is inevitable. Therefore, in this subsection, performance of the proposed classfier is compared with C4.5, as a representative member of the decision tree classifiers. Table 4 lists average classification error rates of the proposed method as well as C4.5 classifier. As indicated by Table 4, except for the Phoneme_CR dataset, the proposed classifier outperforms C4.5.

5. Conclusion

In this study, a simple and computationally efficient local linear neurofuzzy classifier has been introduced, implemented, and tested on a number of well-known datasets. The structure of the antecedent part is obtained during the training phase and is data-driven rather than knowledge based. Input space is first transformed by LDA, so that the linear discriminancy of training samples is maximized. The antecedent structure is then built in the transformed space by axis-orthogonal splits. At each iteration, the local linear classifier with the worst error index is split into two new rules which are then included in the classifier. In addition, the rule consequent parameters are optimized using a local least square approach. The simplicity and speed are the main advantages of the proposed classifier. Together with high performance, this classifier is a good choice for many applications in which the use of more sophisticated classifiers can be impractical.

Acknowledgments

The authors would like to thank M. Ahmadieh and M. Aliyari for their constructive discussions and useful ideas. This research was supported in part by Iran Telecommunication Research Center under Grant T-500-9516.

References

[1] R. Kruse and D. Nauck, "What are neuro-fuzzy classifiers?" in *Proceedings of the 7th International Fuzzy Systems Association World Congress (IFSA '97)*, vol. 3, pp. 228–233, Prague, Czech Republic, June 1997.

[2] E. Czogala and J. Leski, *Fuzzy and Neuro-Fuzzy Intelligent Systems*, Studies in Fuzziness and Soft Computing, Springer, New York, NY, USA, 2000.

[3] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[4] R. Babuška and H. Verbruggen, "Neuro-fuzzy methods for nonlinear system identification," *Annual Reviews in Control*, vol. 27, no. 1, pp. 73–85, 2003.

[5] S. Choudhury, S. Mitra, and S. K. Pal, "Neurofuzzy classification and rule generation of modes of radiowave propagation," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 4, pp. 862–871, 2003.

[6] S. E. Hussein and M. H. Granat, "Intention detection using a neuro-fuzzy EMG classifier," *IEEE Engineering in Medicine and Biology Magazine*, vol. 21, no. 6, pp. 123–129, 2002.

[7] J. S. Taur and C. W. Tao, "A new neuro-fuzzy classifier with application to on-line face detection and recognition," *The Journal of VLSI Signal Processing*, vol. 26, no. 3, pp. 397–409, 2000.

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Wiley-Interscience, New York, NY, USA, 2nd edition, 2001.

[9] D. Nauck and R. Kruse, "NEFCLASS—a neuro-fuzzy approach for the classification of data," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 461–465, Nashville, Tenn, USA, February 1995.

[10] C.-T. Lin, C.-M. Yeh, S.-F. Liang, J.-F. Chung, and N. Kumar, "Support-vector-based fuzzy neural network for pattern classification," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 1, pp. 31–41, 2006.

[11] S. Wu, M. J. Er, and Y. Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 578–594, 2001.

[12] H. Ishibuchi and T. Nakashima, "Effect of rule weights in fuzzy rule-based classification systems," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 506–515, 2001.

[13] K. Nozaki, H. Ishibuchi, and H. Tanaka, "Adaptive fuzzy rule-based classification systems," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 3, pp. 238–250, 1996.

[14] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: survey in soft computing framework," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 748–768, 2000.

[15] F. Hoffmann and O. Nelles, "Genetic programming for model selection of TSK-fuzzy systems," *Information Sciences*, vol. 136, no. 1–4, pp. 7–28, 2001.

[16] O. Nelles, *Nonlinear System Identification from Classical Approaches to Neural Networks and Fuzzy Models*, Springer, New York, NY, USA, 2002.

[17] R. Fullér, *Introduction to Neuro-Fuzzy Systems*, Springer, New York, NY, USA, 2000.

[18] P. McCullagh and J. A. Nelder, *Generalized Linear Models*, Chapman & Hall, London, UK, 2nd edition, 1989.

[19] A. Keles, A. Samet Hasiloglu, A. Keles, and Y. Aksoy, "Neuro-fuzzy classification of prostate cancer using NEFCLASS-J," *Computers in Biology and Medicine*, vol. 37, no. 11, pp. 1617–1628, 2007.

[20] N. Tschichoid-Gürman, "Generation and improvement of fuzzy classifiers with incremental learning using fuzzy RuleNet," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 466–470, Nashville, Tenn, USA, February 1995.

[21] M. Pertselakis, D. Frossyniotis, and A. Stafylopatis, "An adaptable Gaussian neuro-fuzzy classifier," in *Proceedings of Joint International Conference on ICANN/ICONIP*, vol. 2714 of *Lecture Notes in Computer Science*, pp. 925–932, Istanbul, Turkey, June 2003.

[22] S. Abe, M.-S. Lan, and R. Thawonmas, "Tuning of a fuzzy classifier derived from data," *International Journal of Approximate Reasoning*, vol. 14, no. 1, pp. 1–24, 1996.

[23] D. Nauck and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data," *Fuzzy Sets and Systems*, vol. 89, no. 3, pp. 277–288, 1997.

[24] W. Pedrycz and K.-C. Kwak, "The development of incremental models," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 3, pp. 507–518, 2007.

[25] O. Nelles, "Axes-oblique partitioning strategies for local model networks," in *Proceedings of the IEEE International Symposium on Intelligent Control*, vol. 9, pp. 2378–2383, Munich, Germany, October 2006.

[26] H. Abrishami Moghaddam and M. Matinfar, "Fast adaptive LDA using quasi-Newton algorithm," *Pattern Recognition Letters*, vol. 28, no. 5, pp. 613–621, 2007.

[27] H. Abrishami Moghaddam, M. Matinfar, S. M. Sajad Sadough, and Kh. Amiri Zadeh, "Algorithms and networks for accelerated convergence of adaptive LDA," *Pattern Recognition*, vol. 38, no. 4, pp. 473–483, 2005.

[28] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[29] Databases of ELENA project, ftp://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases.

[30] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," Tech. Rep., Department of Information and Computer Science, University of California, Irvine, Calif, USA, 1998, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[31] K. Woods, W. P. Kegelmeyer Jr., and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[32] A. Kostin, "A simple and fast multi-class piecewise linear pattern classifier," *Pattern Recognition*, vol. 39, no. 11, pp. 1949–1962, 2006.

[33] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, Calif, USA, 1993.

[34] S. K. Halgamuge and M. Glesner, "Neural networks in designing fuzzy systems for real world applications," *Fuzzy Sets and Systems*, vol. 65, no. 1, pp. 1–12, 1994.

[35] O. Kaynak, L. A. Zadeh, B. Türksen, and I. J. Rudas, *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*, Springer, New York, NY, USA, 1998.