Expert Systems with Applications 40 (2013) 1451–1466

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Design of face recognition algorithm using PCA -LDA combined for hybrid data pre-processing and polynomial-based RBF neural networks : Design and its application

Sung-Kwun Oh[a,*], Sung-Hoon Yoo[a], Witold Pedrycz[b,c]

[a] *Department of Electrical Engineering, The University of Suwon, Hwaseong-si, Gyeonggi-do, South Korea*
[b] *Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada T6G 2G6*
[c] *Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland*

ARTICLE INFO

Keywords:
PCA, principal component analysis
LDA, linear discriminant analysis
P-RBF NNs, polynomial-based radial basis function neural networks
FCM, Fuzzy C-Means
DE, differential evolution

ABSTRACT

In this study, polynomial-based radial basis function neural networks are proposed as one of the functional components of the overall face recognition system. The system consists of the preprocessing and recognition module. The design methodology and resulting procedure of the proposed P-RBF NNs are presented. The structure helps construct a solution to high-dimensional pattern recognition problems. In data preprocessing part, principal component analysis (PCA) is generally used in face recognition. It is useful in reducing the dimensionality of the feature space. However, because it is concerned with the overall face image, it cannot guarantee the same classification rate when changing viewpoints. To compensate for these limitations, linear discriminant analysis (LDA) is used to enhance the separation between different classes. In this paper, we elaborate on the PCA-LDA algorithm and design an optimal P-RBF NNs for the recognition module.

The proposed P-RBF NNs architecture consists of three functional modules such as the condition part, the conclusion part, and the inference part realized in terms of fuzzy "if–then" rules. In the condition part of fuzzy rules, the input space is partitioned with the use of fuzzy clustering realized by means of the Fuzzy C-Means (FCM) algorithm. In the conclusion part of rules, the connection weight is realized through three types of polynomials such as constant, linear, and quadratic. The coefficients of the P-RBF NNs model are obtained by fuzzy inference method forming the inference part of fuzzy rules. The essential design parameters (including learning rate, momentum, fuzzification coefficient, and the feature selection mechanism) of the networks are optimized by means of differential evolution (DE). The experimental results completed on benchmark face datasets – the AT&T, and Yale datasets demonstrate the effectiveness and efficiency of PCA-LDA combined algorithm compared with other algorithms such as PCA, LPP, 2D-PCA and 2D-LPP. A real time face recognition system realized in this way is also presented.

1. Introduction

In many pattern recognition systems, the paradigm of neural classifiers have been shown to demonstrate many tangible advantages with regard to their learning abilities, generalization aspects, and robustness, (cf. Lawrence, Giles, Tsoi, & Back, 1997; Lin, Kung, & Lin, 1997; Lippman, 1981; Song & Lee, 1998]). Neural networks have been employed and compared to conventional classifiers. The results have shown that the neural networks are effective constructs in comparison with some conventional methods (Lee & Landgrebe, 1997; Zhou, 1999). These properties make neural networks an attractive tool for many pattern classification problems. Among these classifiers, multilayer perceptrons (MLPs) have been in wide use. It is shown that the MLP can be trained to approximate complex discriminant functions (Lippman, 1981). However, the MLP classifier requires a large number of parameters to be determined especially in case of a multilayer topology of this network. The number of iterations required to train networks is often quite high (Patrikar & Provence, 1992). A central issue in neural networks is the problem of learning algorithm. The choice of learning algorithm, network topology, weight initialization and input signal presentations are important factors impacting the learning performance. In particular, the choice of the learning algorithm determines the rate of convergence and the suitability of the solution. Recently radial basis function neural networks (RBF NNs) have been found to be very attractive for many engineering problems. RBF NNs exhibit some advantages including global optimal approximation and classification capabilities, and a rapid conver-

* Corresponding author. Tel.: +82 31 229 8162; fax: +82 31 220 2667.
E-mail addresses: ohsk@suwon.ac.kr (S.-K. Oh), shyoo@suwon.ac.kr (S.-H. Yoo), wpedrycz@ualberta.ca (W. Pedrycz).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.08.046

gence of the learning procedures (see Balasubramanian, Palanivel, & Ramalingam, 2009; Cho & Wang, 1996; Er, Wu, Lu, & Toh, 2002; Mali & Mitra, 2005; Oh, Pderyz, & Park (2004); Park, Oh, & Kim, 2008). In spite of these advantages of RBF NNs, these networks are not free from limitations. In particular, discriminant functions generated by RBF NNs have a relatively simple geometry which is implied by the limited geometric variability of the underlying receptive fields (radial basis functions) located at the hidden layer of the RBF network. Those fields are then aggregated in a linear fashion by the neuron located at the output layer. To overcome limitation of architecture, we propose a concept of the polynomial-based radial basis function neural networks (P-RBF NNs). Given the functional (polynomial) character of their connections in the P-RBF NNs, these networks can generate far more complex nonlinear discriminant functions. Along with the enhanced architectural functionality we consider a comprehensive optimization environment using which this functionality could be fully exploited. With this regard, our intent is to consider using some of the biologically-inspired optimization techniques.

The main objective of this study is to develop an efficient learning algorithm for the P-RBF NNs and its applications in face recognition. The P-RBF NNs is proposed as one of the recognition part of overall face recognition system that consists of two parts such as the preprocessing part and recognition part. The design methodology and the procedure of the proposed P-RBF NNs are presented to construct a solution to high-dimensional pattern recognition problems.

In the data pre-processing part, principal component analysis (PCA) (Gumus, Kilic, Sertbas, & Ucan, 2010; Turk & Pentland, 1991), which is generally used in face recognition, provides high quality features such as a front face. It is useful to express some classes using reduction, since it is effective to maintain the rate of recognition and to reduce the amount of data at the same time. However, because of the use of the entire face image, it cannot retain the same classification rate when dealing with some the change of the viewpoints. It is too difficult to judge whether the changing face image comes from the change of illumination or various facial expression. Thus, to compensate for these limitations, linear discriminant analysis (LDA) (Belhumeur, Hespanha, & Kriegman, 1997; Zhao, Chellappa, & Krishnaswamy, 1998) is used to enhance the separation between classes. However, it still cannot separate the nonlinear data or the classes which have the same average values. In this pre-processing part, we introduce a combination mechanism aggregating PCA and LDA. Then we compare and analyze the performance of the PCA-LDA fusion algorithm.

In the recognition part, we design a P-RBF NNs based on the fuzzy inference mechanism. The essential design parameters of the system are optimized by means of differential evolution. The proposed P-RBF NNs dwell upon structural findings about training data that are expressed in terms of partition matrix resulting from fuzzy clustering in this case being Fuzzy C-Means (FCM). The network is of functional nature as the weights between the hidden layer and the output are treated as some polynomials. The use of the polynomial weights becomes essential in reflecting the nonlinear nature of data encountered in regression or classification problems. From the perspective of linguistic interpretation, the proposed network can be expressed as a collection of "if–then" fuzzy rules. The architecture of the networks discussed here embraces three functional modules reflecting the three phases of input–output mapping realized in rule-based architectures.

This paper is organized as follows: in Section 2, we introduce how to combine PCA-LDA algorithm for extraction the facial features. Section 3 we propose the general architecture of P-RBF NNs. Section 4 discusses the discriminant capabilities of P-RBF NNs and their learning procedure realized by means of the gradient descent method. In Section 5, we discuss the essentials of differential evolution and show its usage in the determination of learning rate, momentum coefficient, fuzzification coefficient and the realization of feature selection. Experimental results and implementation are presented in Section 6. Finally, conclusions are attained in Section 7.

2. Data pre-processing for extraction of facial features

One of the main issues in design of face recognition algorithm is to extract the face features. In this section, a method is proposed for extracting face features from face image. And we introduce how to combine PCA-LDA fusion algorithm.

2.1. Principal component analysis

Principal component analysis (Gumus et al., 2010; Turk & Pentland, 1991) is standard technique used in statistical pattern recognition and signal processing for data reduction and feature extraction (Haykin, 1999). As the pattern often contains redundant information, we map it to a feature space of lower dimensionality.

A face image of size $N \times N$ *pixels* can be considered as a one-dimensional vector of dimensionity N^2. For example, face image from the AT&T (formerly the AT&T database of faces) database of size 112×92 can be considered as a vector of dimension 10,304, or equivalently points in a 10,304 dimensional space. An ensemble of images maps to a collection of points in this highly-dimensional space. The main idea of the principal component is to find the vectors that best account for the distribution of face images within the entire image space. Because these vectors are the eigenvectors of the covariance matrix corresponding to the original face images, and because they are face like in appearance, we refer to them as 'eigenfaces' (see Figs. 1 and 2).

Let the training set of face images be $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$, The average of the set is defined by

$$\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n \tag{1}$$

Each face differs from the average by the vector

$$\Phi_i = \Gamma_i - \Psi \tag{2}$$

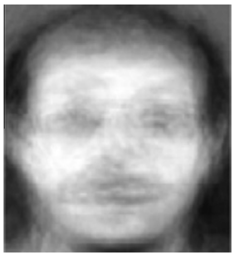


Fig. 1. Average face.

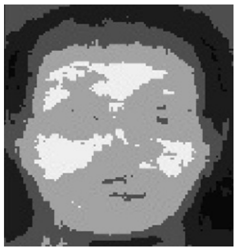


Fig. 2. Eigenface.

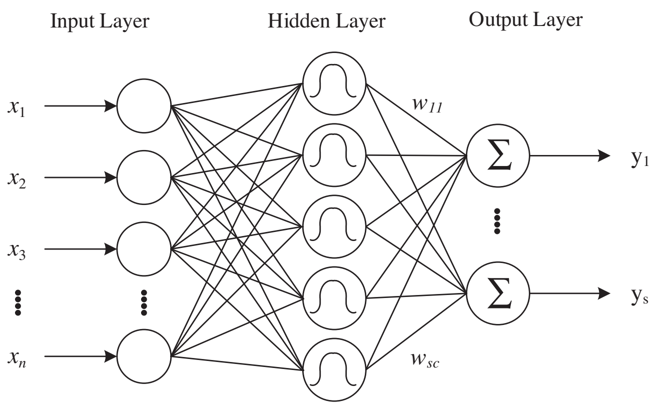


Fig. 3. General architecture of RBF neural networks.

This set of highly dimensional vectors is then subject to principal component analysis, which seeks a set of M orthonormal vectors, U_m, which the best describes the distribution of the data. The kth vector, U_k, is chosen so that

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}\left(U_k^T\Phi_n\right)^2 \tag{3}$$

attains maximum, subject to

$$U_l^TU_k = \delta_{lk} = \begin{Bmatrix} 1, & \text{if } l = k \\ 0, & \text{otherwise} \end{Bmatrix} \tag{4}$$

The vectors Uk are the eigenvectors of the covariance matrix

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T \tag{5}$$

where the matrix $A = [\Phi_1\ \Phi_2 \ldots \Phi M]$. The covariance matrix C, however is $N^2 \times N^2$ real symmetric matrix, and calculating the N^2 eigenvectors and eigenvalues is an intractable task for typical image sizes. We need a computationally feasible method to find these eigenvectors.

Consider the eigenvectors vi of A^TA such that

$$A^TAv_i = \mu_i v_i \tag{6}$$

After multiplying both sides by A, we have

$$AA^TAv_i = \mu_i Av_i \tag{7}$$

We see that Av_i are the eigenvectors and μ_i are the eigenvalues of $C = AA^T$.

Following these analysis, we construct the $M \times M$ matrix $L = A^TA$, where $L_{mn} = \Phi_m^T\Phi_n$, and find the M eigenvectors, v_l, of L. There vectors determine linear combinations of the M training set face images to form the eigenfaces U_l.

$$U_l = \sum_{k=1}^{M} v_{lk}\Phi_k, \quad l = 1, \ldots, M \tag{8}$$

With this analysis, the calculations are greatly reduced, from the order of the number of pixels in the images (N^2) to the order of the number of images in the training set (M). In practice, the training set of face images will be relatively small ($M \ll N^2$), and the calculations become quite manageable. The associated eigenvalues allow us to rank the eigenvectors according to their usefulness when characterizing the variation among the images. The eigenface images calculated from the eigenvectors of L span a basis set that can be used to describe face images. (Sirovich & Kirby, 1987, 1990) evaluated a limited version of this framework on an ensemble of 115 images ($M = 115$) images of Caucasian males digitized in a controlled manner, and found that 40 eigenfaces ($M' = 40$) were sufficient for a very good description of face images. In practice, a smaller M' can be sufficient for identification, since accurate reconstruction of the image is not a requirement. In the framework of face recognition, the operation is a pattern recognition task rather than image reconstruction. The eigenfaces span an M' dimensional subspace of the original N^2 image space and hence, the M' significant eigenvectors of the L matrix with the largest associated eigenvalues, are sufficient for reliable representation of the faces in the face space characterized by the eigenfaces.

Then a new face image (Γ) is transformed into its eigenface components (projected onto "face space") by a simple operation,

$$w_k = U_k^T(\Gamma - \Psi) \tag{9}$$

for $k = 1, \ldots, M'$. The weights form a projection vector, describing the contribution of each eigenface in representing the input face image, treating the eigenfaces as a basis set for face images. The projection vector is then used in a standard pattern recognition algorithm to identify which of a number of predefined face classes, if any, best describes the face.

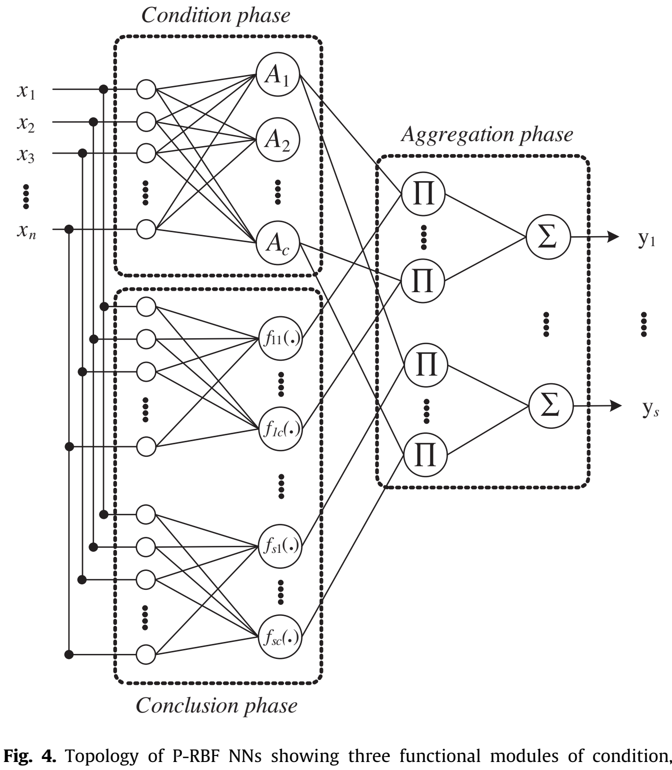


Fig. 4. Topology of P-RBF NNs showing three functional modules of condition, conclusion and aggregation phases.

2.2. Linear discriminant analysis

Linear discriminant analysis or Fisherfaces method (Belhumeur et al., 1997) overcomes the limitations of eigenfaces method by applying the Fisher's linear discriminant criterion. This criterion tries to maximize the ratio of the determinant of the between-class scatter matrix of the projected samples to the determinant of the within-class scatter matrix of the projected samples. Fisher discriminants group images of the same class and separates images of different classes. Images are projected from N^2-dimensional space to C dimensional space (where C is the number of classed of images). For example, consider two sets of points in 2-dimensional space that are projected onto a single line. Depending on the direction of the line, the points can either be mixed together or separated. Fisher discriminants find the line that best separated the points. Unlike the PCA method that extracts features to best

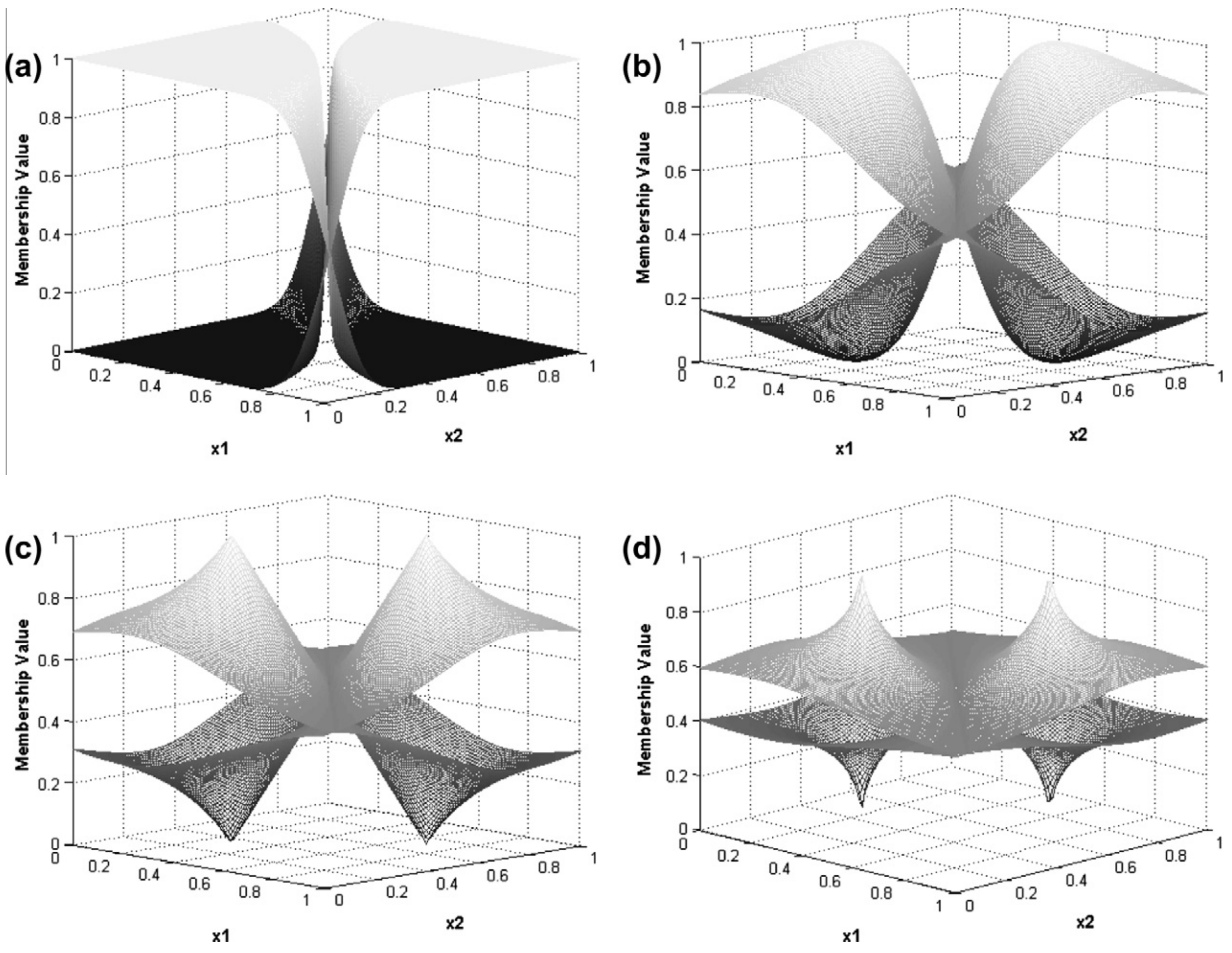


Fig. 5. The shape of membership function formed by the FCM algorithm for selected values of the fuzzification coefficient: (a) m = 1.1, (b) m = 2.0, (c) m = 3.0, (d) m = 4.0.

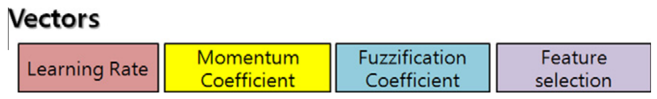


Fig. 6. Construction of vectors of parameters.

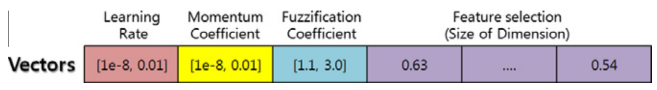


Fig. 7. The structure of parameter vectors for optimization of P-RBF NNs.

represent face images; the LDA method tries to find the subspace that best discriminates different face classes. The within-class scatter matrix, also called intra-personal, represents variations in appearance of the same individual due to different lighting and face expression. While the between-class scatter matrix, also called the extra-personal, represents variations in. On the one hand, we maximize the distance between the face images of different classes. On the other hand, we minimize the distance between the face images of the same class. In another words, the objective is to maximize the between-class scatter S_B, while minimizing the within-class scatter matrix S_W in the projective subspace.

The within-class scatter matrix S_W and the between-class scatter matrix S_B are defined as

$$S_W = \sum_{j=1}^{C}\sum_{i=1}^{N_j}\left(\Gamma_i^j - \mu_j\right)\left(\Gamma_i^j - \mu_j\right)^T \tag{10}$$

where Γ_i^j is the ith sample of class j, μ_j is the mean of class j, C is the number of classes, N_j is the number of samples in class j.

$$S_B = \sum_{j=1}^{C}(\mu_j - \mu)(\mu_j - \mu)^T \tag{11}$$

Here μ represents the mean of all classes. The subspace for LDA is spanned by a set of vectors $W = [W1, W2, \ldots, Wd]$, satisfying

$$W = \arg\max = \left|\frac{W^T S_B W}{W^T S_W W}\right| \tag{12}$$

The within class scatter matrix expresses how face images are distributed closely within classes while the between class scatter matrix quantifies how classes are separated from each other. When face images are projected onto the discriminant vectors W, face images should be distributed closely within classes and be separated between classes, as much as possible. In other words, these discriminant vectors minimize the denominator and maximize the numerator in (12). W can therefore be constructed with the aid of the eigenvectors of $S_w^{-1}S_B$.

2.3. A fusion of PCA and LDA

As mentioned above, both the PCA and LDA are the classical algorithms. They are widely used in the field of classification and many approaches using PCA or LDA have been reported, respectively.

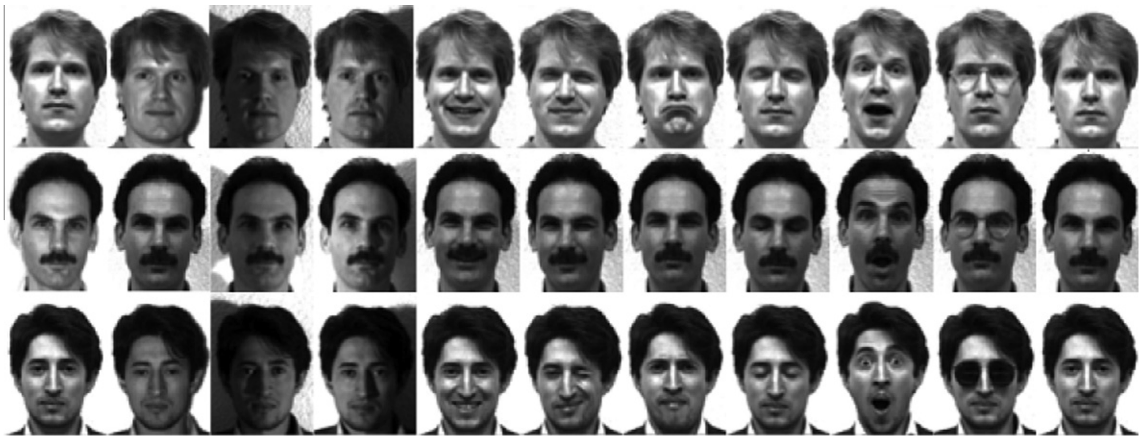


Fig. 8. Samples of face images in the Yale face database.

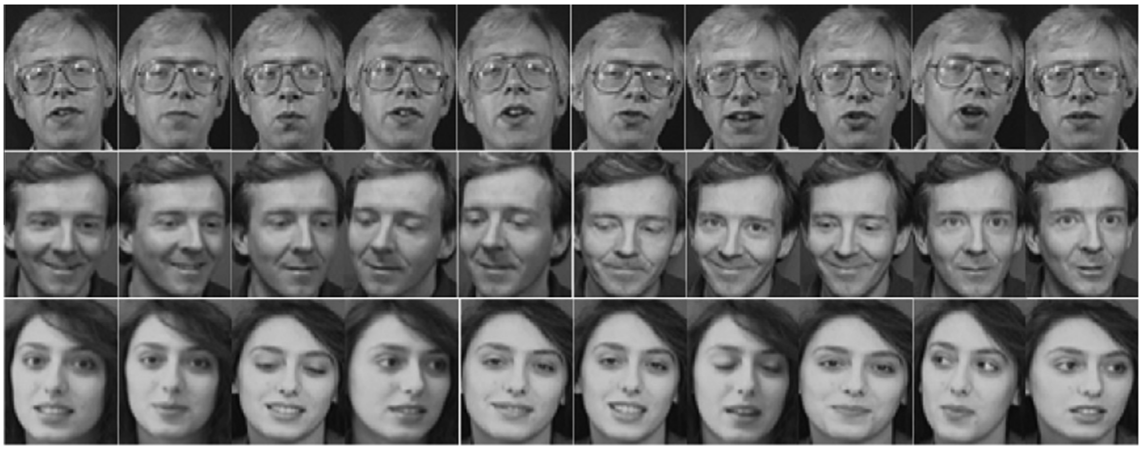


Fig. 9. Samples of face images in the AT&T database.

Table 1
Description of face image database used for experiment.

Yale database	It contains 165 face images of 15 individuals. There are 11 images per subject, one for each facial expression or configuration: center-light, glasses/no glasses, happy, normal, left-light, right-light, sad, sleepy, surprised and wink. Samples of Yale face database are shown in Fig. 8.
AT&T database	It contains 400 face images from 40 individuals in different states. The total number of images for each person is 10. None of the 10 samples is identical to any other sample. They vary in position, rotation, scale and expression. The changes in orientation have been accomplished by rotating the person a maximum of 20° in the same plane; also each person has changed his/her facial expression in each 10 samples (open/close eye, smiling/not smiling). The changes in scale have been achieved by changing the distance between the person and the video camera. For some individuals, the images were taken at different times, varying facial details (glasses/no glasses). Each image was digitized and presented by a 112 × 92 pixel array whose gray levels between 0 and 255. Samples of the AT&T database are shown in Fig. 9.

Based on this observation, we expected that the approach combining classifiers leads to better results.

Let $A = \{a^1, a^2, a^3, \ldots, a^n\}$ be a set of n patterns, where each of them is described by N features and belongs to one of the R classes (D_i, $i = 1, 2, \ldots, R$). Let l_i denote the number of patterns in the ith class. The vector a denotes a point in an N dimensional space. The vectors of n patterns are arranged column-wise to form the data matrix A of dimensionality $N \times n$.

$$A = [a^1, \ldots, a^n] \tag{13}$$

Let B_{ij} ($j = 1, 2, \ldots, l_i$) be the vectors of the ith class. Let $\overline{A}$ be the overall mean vector of the matrix A and let $\overline{B_i}$ be the mean vector of the ith class, i.e.,

$$\overline{A} = \frac{1}{n}\sum_{k=1}^{n} a^{(k)}$$

$$\overline{B_i} = \frac{1}{l_i}\sum_{j=1}^{l_i} B_{ij} \tag{14}$$

The vectors of the data matrix A are centered by subtracting the mean vector $\overline{A}$ from all the column vectors of A to get the covariance matrix, as in principal component analysis, say $C^1(N \times N)$ of the column vectors, which is given by

$$C^1 = [A - \overline{A}][A - \overline{A}]^T \tag{15}$$

Similarly, as in linear discriminant analysis the covariance matrix, say C^2 (of dimensionality $N \times N$) is given by

Table 2
Parameters of DE for the optimization of P-RBF NNs.

P-RBF NNs		
Number of learning iterations		100
Polynomial type		Linear, reduced quadratic
Data split		Training : Validation : Testing = 5 : 3 : 2
Differential evolution		
Number of generations		20
Number of populations		50
F weight		0.95
Crossover		0.2
The range of Search space	Learning rate	[1e−8,0.01]]
	Momentum coefficient	[1e−8,0.01]
	Fuzzification coefficient	[1.1,3.0]
	Feature selection	More than 0.45

Table 3
Image size and division of Yale face dataset.

Database	Image size	Division of dataset			
		Total	Training	Validation	Testing
Yale	243 × 320	165	90	45	30

Table 4
Classification performance on Yale face dataset using PCA method.

Classifier model	Number of rules	Polynomial type	Classification rate (%)		
			Training AVG ± STD	Validation AVG ± STD	Testing AVG ± STD
DE- pRBF NNs (*Case 1*)	2	L-RBF NNs	99.13 ± 2.09	96.67 ± 2.22	91.11 ± 4.80
		RQ-RB FNNs	99.45 ± 1.09	93.89 ± 3.34	90.55 ± 4.21
	3	L-RBF NNs	99.09 ± 1.13	95.56 ± 3.14	93.34 ± 6.29
		RQ-RBF NNs	99.42 ± 1.17	93.89 ± 2.13	90.57 ± 5.31
	4	L-RBF NNs	**99.45 ± 1.11**	**96.11±3.58**	**93.45±2.87**
		RQ-RBF NNs	99.72 ± 0.57	93.89 ± 2.79	88.34 ± 2.13
	5	L-RBF NNs	98.89 ± 2.22	96.67 ± 2.87	90.45 ± 3.95
		RQ-RBF NNs	99.67 ± 0.67	92.78 ± 2.79	87.78 ± 6.91
	6	L-RBF NNs	99.63 ± 0.64	97.78 ± 3.85	92.59 ± 3.39
		RQ-RBF NNs	99.21 ± 1.37	95.55 ± 3.85	91.85 ± 4.63

$$C^2 = S_W^{-1} S_B \tag{16}$$

where S_B and S_W are the between class and the within class scatter matrices, respectively being expressed as,

$$S_B = \frac{1}{n}\sum_{i=1}^{R} l_i(\overline{B_i} - \overline{A})(\overline{B_i} - \overline{A})^T \tag{17}$$

$$S_W = \frac{1}{n}\sum_{i=1}^{R}\sum_{k=1}^{l_i}(B_{ik} - \overline{B_i})(B_{ik} - \overline{B_i})^T \tag{18}$$

In the PCA and LDA approaches, the eigenvectors of the covariance matrix C^1 and C^2 are computed to project the data onto the eigen space, respectively. In this work, we combine the two covariance matrix say C^{Fused} which is defined as

$$C^{Fused} = \Im(C^1, C^2) \tag{19}$$

Table 5
Classification performance on Yale face dataset using PCA-LDA fusion method.

Classifier model	Number of rules	Polynomial type	Classification rate (%)		
			Training AVG ± STD	Validation AVG ± STD	Testing AVG ± STD
DE-pRBF NNs (*Case 2*)	2	L-RBF NNs	**99.56±0.99**	**98.56 ± 1.99**	**95.57 ± 2.49**
		RQ-RBF NNs	99.73 ± 0.65	97.29 ± 2.91	93.61 ± 3.66
	3	L-RBF NNs	98.85 ± 1.93	99.78 ± 2.06	95.13 ± 4.80
		RQ-RBF NNs	99.47 ± 0.73	98.67 ± 1.99	92.02 ± 4.61
	4	L-RBF NNs	99.73 ± 0.59	98.22 ± 2.89	94.67 ± 3.37
		RQ-RBF NNs	99.28 ± 1.04	94.67 ± 3.97	93.44 ± 2.53
	5	L-RBF NNs	99.73 ± 0.78	99.33 ± 1.99	95.11 ± 3.98
		RQ-RBF NNs	99.07 ± 5.84	98.67 ± 1.52	92.25 ± 3.65
	6	L-RBF NNs	99.50 ± 1.79	98.89 ± 2.10	94.27 ± 3.72
		RQ-RBF NNs	99.73 ± 0.59	98.67 ± 2.26	92.89 ± 4.54

Table 6
Comparison of the average recognition rate for several models on the Yale face dataset.

	Recognition rate (%)	Reference	Year (fcv)
LPP	87.7	He and Niyogi (2003)	2003
2D-PCA	86.7	Cai et al. (2007)	2004
2D-LPP	88.1	Yang et al. (2004)	2007
SR	84.9	Hu et al. (2007)	2007
Proposed method (*Case 1*)	94.5	This study	5-fcv
Proposed method (*Case 2*)	95.6	This study	5-fcv

Here $\Im$ is a fusing function, which integrates the two covariance matrices together into a single covariance matrix. The fused matrix is used for computing eigenvectors for data projection.

In this study, to make that selection if scalars (weight factors) a non-parametric, we recommend to consider the elements of the covariance matrix of another classifier as scalars (weight factors) for the respective elements of the covariance matrix of the original classifier. The advantage of this way of treating elements of the covariance matrix as weight factors of the other covariance matrix is that the weight factors are themselves derived from the sample data, there by expecting an enhance in the classification accuracy.

$$C^{Fused} = \Im(C^1, C^2) = C^1 \circ C^2 = C^1_{ij} \times C^2_{ij} \tag{20}$$

where C_{ij} is the (i,j)th value of the covariance matrix.

The fused covariance matrix C^{Fused} is expected to preserve the nature of both C^1 and C^2 covariance matrices thereby increasing the performance of the model.

The eigenvectors of the fused covariance matrix C^{Fused} are computed and sorted according to their respective eigenvalues. Let $V = \{v_1, v_2, v_3, \ldots, v_N\}$ be the eigenvalues such that $v_1 \geqslant v_2 \geqslant \cdots \geqslant v_N$. Let $X = [X^1, \ldots, X^N]$ be the sorted sequence of eigenvectors. The data matrix A is then projected onto the eigenspace using X to get $P(N \times n)$ consisting n columns, where

$$P = X^T A \tag{21}$$

and each columns in P describes a transformed point in the N dimensional eigenspace. The advantage of transforming the original pattern vectors into the eigenspace is that the dimensionality required to represent the pattern can be reduced to $m \ll N$, that is, the top m coordi-

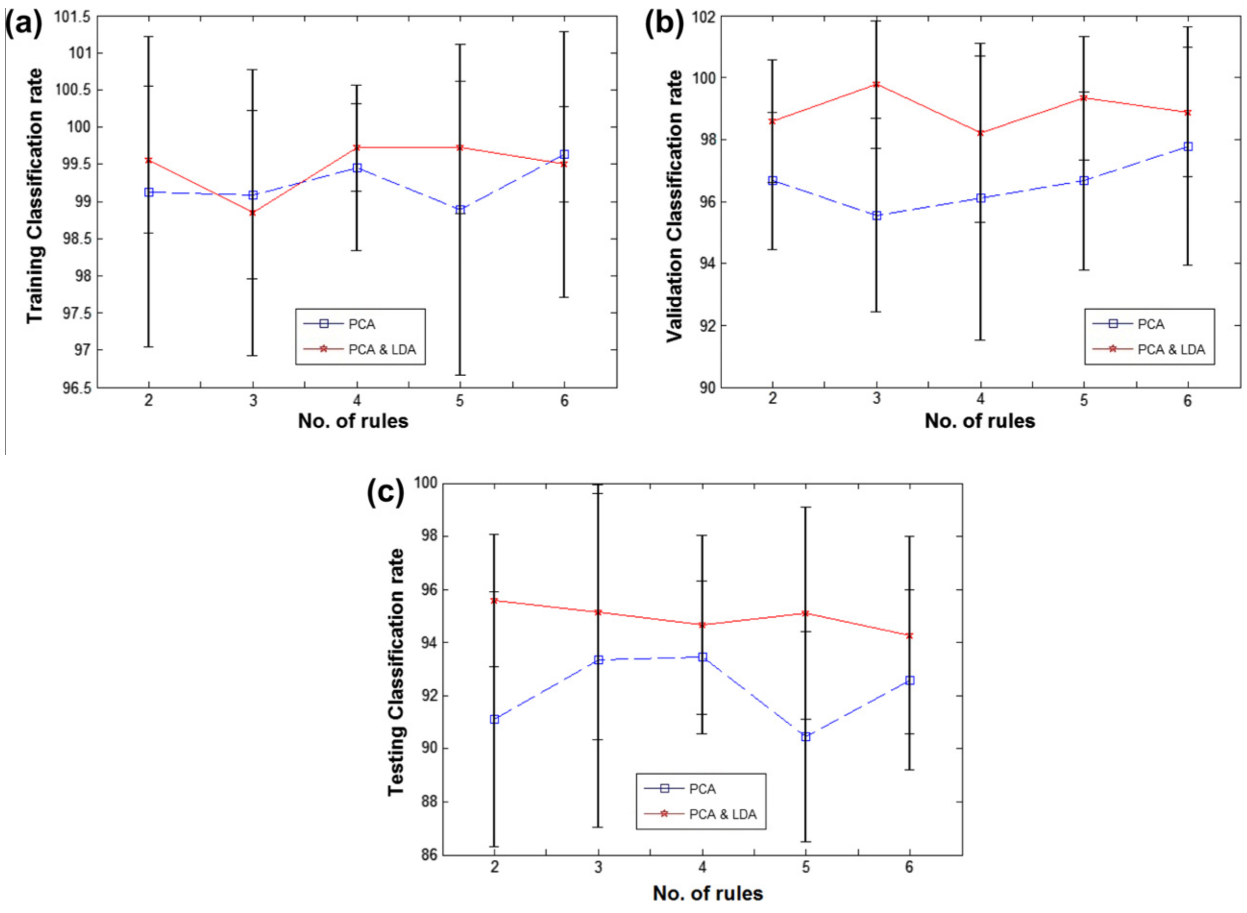


Fig. 10. Comparison of classification rate according to different feature extraction method (polynomial type: linear): (a) training data (b) validation data (c) testing data.

nate points of a transformed column vector in P matrix would represent the pattern with little or no loss of information.

3. Polynomial-based radial basis function networks: a general topology

In what follows, we discuss a general topology of polynomial-based radial basis function networks (P-RBF NNs).

3.1. Architecture of polynomial-based RBF networks

The general architecture of the radial basis function neural networks (RBF NNs) consists of three layers as shown in Fig. 3.

RBF NNs has a single hidden layer. Each node in the hidden layer expresses a level of activation of the receptive field (radial basis function) $\Theta(\mathbf{x})$ given some incoming input $\mathbf{x}$. The j^{th} output $y_j(\mathbf{x})$ is a weighted linear combination of the activation levels of the receptive fields:

$$y_j(\mathbf{x}) = \sum_{i=1}^{c} w_{ji}\Theta_i(\mathbf{x}) \tag{22}$$

$j = 1, \ldots, s$. 's' stands for the number of output (the number of classes encountered in classification problems). In the case of the Gaussian type of RBFs, we have

$$\Theta_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{v}_i\|^2}{2\sigma_i^2}\right) \tag{23}$$

where $\mathbf{x}$ is the n-dimensional input vector $[x_1, \ldots x_n]^T$, $\mathbf{v}_i = [v_{i1}, \ldots v_{in}]^T$ is the center of ith basis function $\Theta_i(\mathbf{x})$ and c is the number of the nodes of the hidden layer. Typically the distance $\|.\|$ used in (23) is the Euclidean one (Staiano, Tarliaferri, & Pedrycz, 2006).

The proposed P-RBF NNs exhibits a similar topology as the one encountered in RBF NNs. However the functionality and the associated design process exhibit some evident differences. In particular, the receptive fields do not assume any explicit functional form (say, Gaussian, ellipsoidal, etc.), but are directly reflective of the nature of the data and come as the result of fuzzy clustering. Given the prototypes formed by the FCM method (Aiver, Pyun, Huang, O'Brien, & Gray, 2005; Bezdek, 1981), the receptive fields are expressed in the following way

$$\Theta_i(\mathbf{x}) = A_i(\mathbf{x}) = \frac{1}{\sum_{j=1}^{c}\left(\frac{\|\mathbf{x}-\mathbf{v}_i\|^2}{\|\mathbf{x}-\mathbf{v}_j\|^2}\right)} \tag{24}$$

In the addition, the weights between the output layer and the hidden layer are not single numeric values but come in the form of polynomials of input variables (hence the term of functional links that becomes present in this architecture)

$$w_{ji} = f_{ji}(\mathbf{x}) \tag{25}$$

The neuron positioned at the output layer realizes a linear combination of the activation levels of the corresponding receptive fields hence (22) can be rewritten as follows,

$$y_j(\mathbf{x}) = \sum_{i=1}^{c} f_{ji}(\mathbf{x})A_i(\mathbf{x}) \tag{26}$$

The above structure of the classifier can be represented through a collection of fuzzy rules

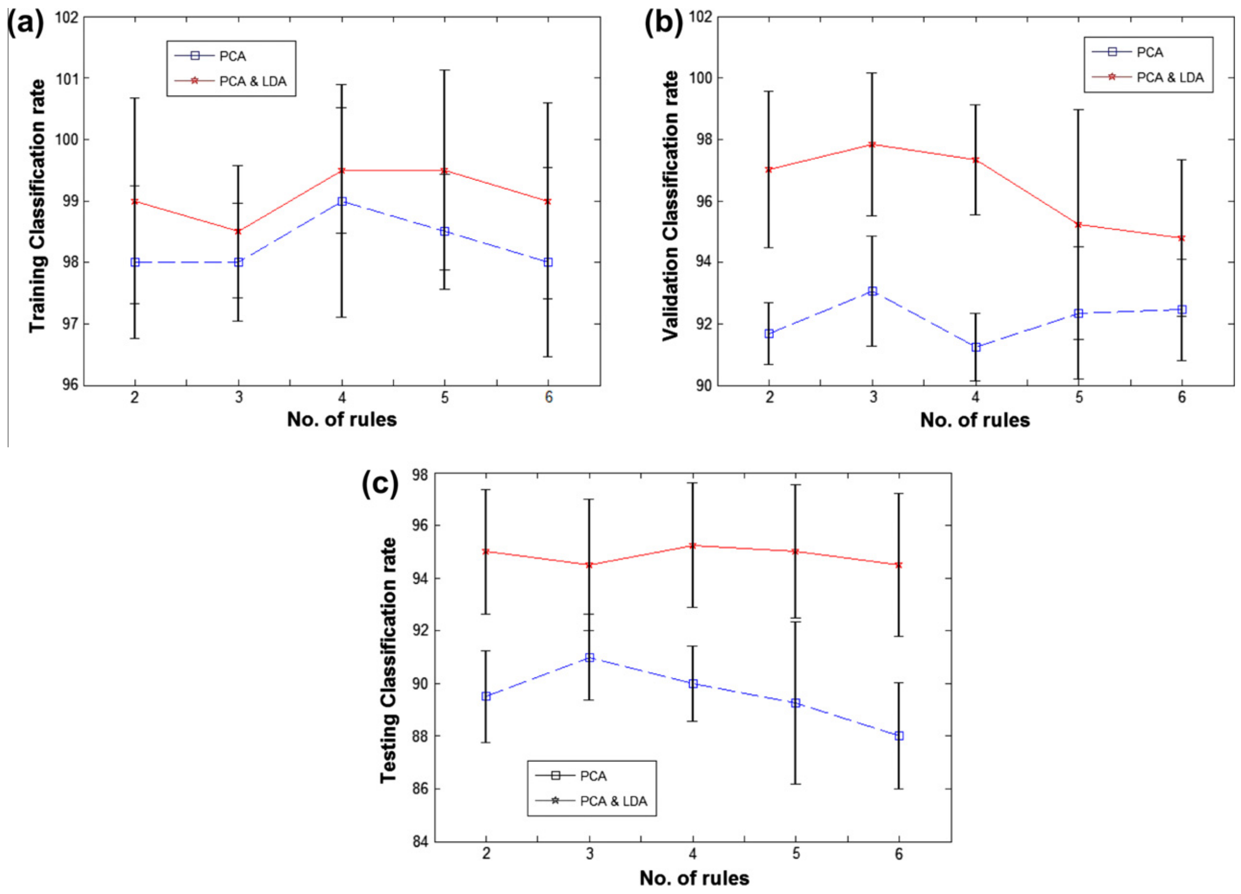


Fig. 11. Comparison of classification rate according to different feature extraction method (polynomial type: linear): (a) training data (b) validation data (c) testing data.

Table 7
Image size and division of AT&T dataset.

Database	Image size	Division of dataset			
		Total	Training	Validation	Testing
AT&T	92 × 112	400	200	120	80

$$\text{If } \mathbf{x} \text{ is } A_i \text{ then } f_{ji}(\mathbf{x}) \tag{27}$$

where, the family of fuzzy sets A_i is the i-cluster (membership function) of the ith fuzzy rule, $f_{ji}(\mathbf{x})$ is a polynomial function generalizing a numeric weight used in the standard form of the RBF NNs, and c is the number of fuzzy rules (clusters), and $j = 1, \ldots, s$; 's' is the number of output.

3.2. Three processing phases of polynomial-based RBF neural networks

The proposed P-RBF NN is implemented by realizing three processing phases that is, condition, conclusion and aggregation phases. The condition and conclusion phases relate to the formation of the fuzzy rules and their ensuing analysis. The Aggregation phase is concerned with a fuzzy inference (mapping procedure).

3.2.1. Condition phase of networks

The condition phase of P-RBF NNs is handled by means of the Fuzzy C-Means. In this section, we briefly review the objective function-based fuzzy clustering with intent of highlighting it key features pertinent to this study. The FCM algorithm is aimed at the formation of 'c' fuzzy sets (relations) in $\mathbf{R}^n$. The objective function Q guiding the clustering is expressed as a sum of the distances of individual data from the prototypes $\mathbf{v}_1, \mathbf{v}_2, \ldots,$ and $\mathbf{v}_c$,

$$Q = \sum_{i=1}^{c}\sum_{k=1}^{N} u_{ik}^{m}\|\mathbf{x}_k - \mathbf{v}_i\|^2. \tag{28}$$

Here, $\|\ \|$ denotes a certain distance function; 'm' stands for a fuzzification factor (coefficient), $m > 1.0$. N is the number of patterns (data). The resulting partition matrix is denoted by $U = [u_{ik}]$, $i = 1, 2, \ldots, c$; $k = 1, 2, \ldots, N$. While there is a substantial diversity as far as distance functions are concerned, here we adhere to a weighted Euclidean distance taking on the following form

$$\|\mathbf{x}_k - \mathbf{v}_i\|^2 = \sum_{j=1}^{n}\frac{(x_{kj} - v_{ij})^2}{\sigma_j^2} \tag{29}$$

with σ_j being a standard deviation of the jth variable. While not being computationally demanding, this type of distance is still quite flexible and commonly used.

Consider the set X, which consists of N patterns treated as vectors located in some n-dimensional Euclidean space, that is, $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_k \in \mathbf{R}^n$, $1 \leqslant k \leqslant N$. In clustering we assign patterns $\mathbf{x}_k \in$ X into c clusters, which are represented by its prototypes $\mathbf{v}_i \in \mathbf{R}^n$, $1 \leqslant i \leqslant c$. The assignment to individual clusters is expressed in terms of the partition matrix U = [u$_{ik}$] where

$$\sum_{i=1}^{c} u_{ik} = \mathbf{1}, \quad 1 \leqslant k \leqslant N \tag{30}$$

and

$$0 < \sum_{k=1}^{N} u_{ik} < N, \quad 1 \leqslant i \leqslant c \tag{31}$$

Table 8
Classification performance on AT&T dataset using PCA method.

Classifier model	Number of rules	Polynomial type	Classification rate (%)		
			Training AVG ± STD	Validation AVG ± STD	Testing AVG ± STD
DE-pRBF NNs (*Case 1*)	2	L-RBF NNs	98.50 ± 1.24	91.68 ± 1.02	89.50 ± 1.75
		RQ-RBF NNs	99.50 ± 1.27	89.75 ± 3.57	89.25 ± 3.97
	3	L-RBF NNs	**98.00** ± 0.96	**93.06** ± 1.79	**91.00** ± 1.62
		RQ-RBF NNs	99.00 ± 1.15	92.92 ± 2.62	87.00 ± 2.17
	4	L-RBF NNs	98.50 ± 1.89	91.24 ± 1.09	90.00 ± 1.42
		RQ-RBF NNs	99.50 ± 2.12	92.29 ± 2.31	86.50 ± 4.13
	5	L-RBF NNs	98.00 ± 0.93	92.35 ± 2.16	89.25 ± 3.08
		RQ-RBF NNs	99.00 ± 1.24	91.93 ± 2.23	87.50 ± 5.36
	6	L-RBF NNs	98.00 ± 1.54	92.46 ± 1.65	88.00 ± 2.02
		RQ-RBF NNs	99.50 ± 1.67	88.50 ± 3.23	85.75 ± 4.48

Table 9
Classification performance on AT&T dataset using PCA-LDA fusion method.

Classifier model	Number of rules	Polynomial type	Classification rate (%)		
			Training AVG ± STD	Validation AVG ± STD	Testing AVG ± STD
DE-pRBF NNs (*Case 2*)	2	L-RBF NNs	99.00 ± 1.67	97.02 ± 2.55	95.00 ± 2.37
		RQ-RBF NNs	99.50 ± 1.22	95.58 ± 3.06	93.00 ± 3.69
	3	L-RBF NNs	98.50 ± 1.08	97.83 ± 2.32	94.50 ± 2.50
		RQ-RBF NNs	99.00 ± 2.32	95.06 ± 4.20	94.25 ± 2.44
	4	L-RBF NNs	**99.50 ± 1.02**	**97.35 ± 1.79**	**95.25 ± 2.36**
		RQ-RBF NNs	99.50 ± 1.99	96.88 ± 3.49	94.50 ± 3.90
	5	L-RBF NNs	99.50 ± 1.63	95.22 ± 3.74	95.00 ± 2.53
		RQ-RBF NNs	98.50 ± 1.91	95.56 ± 3.58	93.50 ± 4.67
	6	L-RBF NNs	99.00 ± 1.60	94.79 ± 2.54	94.50 ± 2.70
		RQ-RBF NNs	99.00 ± 0.98	93.11 ± 3.11	94.00 ± 3.53

Table 10
Comparison of the average recognition rate for several models on the AT&T dataset.

	Recognition rate (%)	Reference	Year (fcv)
LPP	87.4	He and Niyogi (2003)	2003
2D-PCA	92.5	Cai et al. (2007)	2004
2D-LPP	94.2	Yang et al. (2004)	2007
SR	91.6	Hu et al. (2007)	2007
Proposed method (*Case* 1)	91.0	This study	5-fcv
Proposed method (*Case* 2)	95.0	This study	5-fcv

The minimization of Q is realized in successive iterations by adjusting both the prototypes and entries of the partition matrix, that is min $Q(U,\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_c)$. The corresponding formulas used in an iterative fashion read as follows

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left(\frac{\|\mathbf{x}_k-\mathbf{v}_i\|}{\|\mathbf{x}_k-\mathbf{v}_j\|}\right)^{\frac{2}{m-1}}}, \quad 1 \leqslant k \leqslant N,\ 1 \leqslant i \leqslant c \tag{32}$$

and

$$\mathbf{v}_i = \frac{\sum_{k=1}^{N} u_{ik}^m \mathbf{x}_k}{\sum_{k=1}^{N} u_{ik}^m}, \quad 1 \leqslant i \leqslant c \tag{33}$$

The properties of the optimization algorithm are well documented in the literature (cf. Aiver et al., 2005). In the context of our investigations, we note that the resulting partition matrix produces '*c*' fuzzy relations (multivariable fuzzy sets) with the membership functions $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_c$ forming the corresponding rows of the partition matrix U, that is $U = [\mathbf{u}_1^T \mathbf{u}_2^T \ldots \mathbf{u}_c^T]$. From the design standpoint, there are several essential parameters of the FCM that impacts the usage of the produced results. These parameters concern the number of clusters, the values of the fuzzification coefficient and a form of the distance function. The fuzzification coefficient exhibits a significant impact on the form (shape) of the developed clusters. The commonly used value of "*m*" is equal to 2. Lower values of the fuzzification coefficient produce more Boolean-like shapes of the fuzzy sets where the regions of intermediate membership values are very much reduced. When we increase the values of "*m*" above 2, the resulting membership functions start to become "spiky" with the values close to 1 in a very close vicinity of the prototypes as shown in Fig. 5.

As shown in Fig. 5, the shape of membership function is directly affected by the values of the fuzzification coefficient '*m*'. Given this observation, we anticipate that the adjustment of the values of *m* will substantially impact the performance of the network (see Fig. 6).

3.2.2. Conclusion part of the network

Polynomial functions are dealt with in the conclusion phase. For convenience, we omit the suffix *j* from $f_{ji}(\mathbf{x})$ shown in Fig. 5 and (27). Several classes of polynomials are worth noting

$$\text{Constant; } f_i(\mathbf{x}) = a_{i0} \tag{34}$$

$$\text{Linear; } f_i(\mathbf{x}) = a_{i0} + \sum_{j=1}^{n} a_{ij}x_j \tag{35}$$

$$\text{Quadratic; } f_i(\mathbf{x}) = a_{i0} + \sum_{j=1}^{n} a_{ij}x_j + \sum_{j=1}^{n}\sum_{k=j}^{n} a_{ijk}x_jx_k \tag{36}$$

These functions are activated by partition matrix and lead to local regression models located at the condition phase of the individual rules.

In case of the quadratic function, the dimensionality of the problem increases quite quickly especially when dealing with problems of high dimensionality. The reduced quadratic function

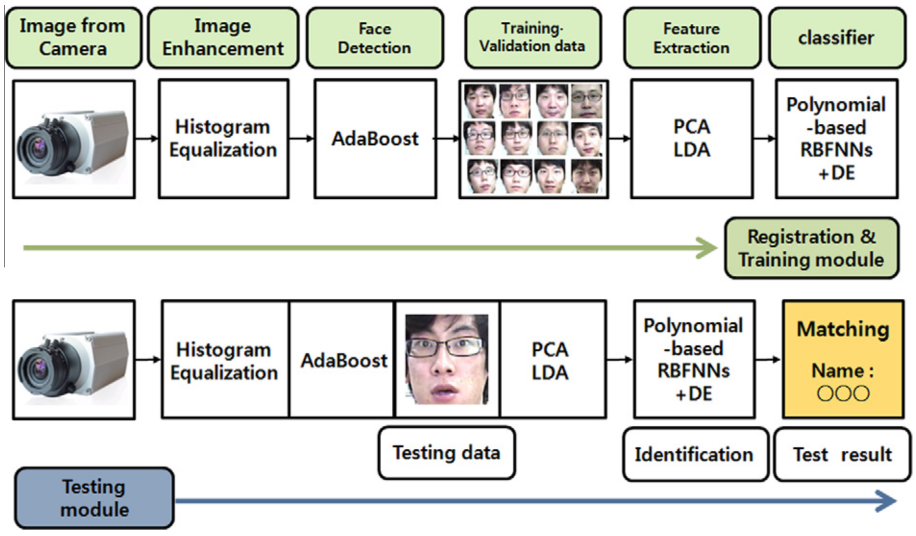


Fig. 12. Overall processing realized in the face recognition system.

$$f_i(\mathbf{x}) = a_{i0} + \sum_{j=1}^{n} a_{ij}x_j + \sum_{k=1}^{n} a_{ijk}x_k^2 \tag{37}$$

is also discussed with an intent of reducing computational burden.

The use of some constant in (34) (which are special cases of the polynomial functions) reduces the P-RBF NNs to the standard RBF neural networks as illustrated in Fig. 3.

3.2.3. Aggregation phase of networks

Let us consider the P-RBF NNs structure by considering the fuzzy partition realized in terms of FCM as shown in Fig. 4. The node denoted by Π is realized as a product of the corresponding fuzzy set and the polynomial function. The family of fuzzy sets A_i forms a partition (so that the sum of membership grades sum up to one at each point of the input space). The "$\sum$" neuron is described by a linear sum as shown in (26). The output of P-RBF NNs can be obtained by following a standard inference mechanism used in rule-based systems (Bezdek, 1981),

$$y_j = g_j(\mathbf{x}) = \sum_{i=1}^{c} \frac{u_i f_{ji}(\mathbf{x})}{\sum_{k=1}^{c} u_k} = \sum_{i=1}^{c} u_i f_{ji}(\mathbf{x}) \tag{38}$$

where $u_i = A_i(\mathbf{x})$. All the entries sum up to 1 as indicated by (30). $g_j(\mathbf{x})$ describes here the discriminant function used for discerning jth class.

Based on the local polynomial-like representation, the global characteristics of the P-RBF NNs result through the composition of their local relationships.

4. Polynomial-based radial basis function neural networks classifiers and its learning using gradient descent method

In this section, we consider the use P-RBF NNs as classifiers and show how their functionality gives rise to highly nonlinear discriminant functions.

4.1. The discriminant function

There are many different ways to describe pattern classifiers. One of the most useful ways is the one realized in terms of a set of discriminant functions $g_i(\mathbf{x})$, $i = 1, \ldots, m$ (where m stands for the number of classes). The classifier is said to assign a input vector $\mathbf{x}$ to class ω_i if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \text{ for all } j \neq i \tag{39}$$

Thus, the classifiers are viewed as networks that compute m discriminant functions and select the category corresponding to the largest value of the discriminant.

In this paper, the proposed P-RBF NNs classifier is used as a discriminate function for two-class or multi-class. If a classification problem is multi-class one then we use (39) as discriminant function, otherwise, we use the following decision rule defined commonly as a single discriminant function $g(\mathbf{x})$ in two-class problem.

$$\text{Decide } \omega_1 \text{ if } g(\mathbf{x}) > 0; \text{ otherwise decide } \omega_2 \tag{40}$$

The final output of networks, (38), is used as a discriminant function $g(\mathbf{x})$ and can be rewritten in a form of the linear combination

$$g(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\mathbf{fx} \tag{41}$$

where $\mathbf{a}$ is a vector of coefficients of the polynomials used in the conclusion part of the rules in (34)–(37) and $\mathbf{fx}$ is a matrix of U and $\mathbf{x}$. These can be defined for each of polynomial as follows.

(i) Constant;

$$\mathbf{a}^T = [a_{10}, \ldots, a_{c0}], \mathbf{fx} = [u_1, \ldots, u_c]^T$$

(ii) Linear;

$$\mathbf{a}^T = [a_{10}, \ldots, a_{c0}, a_{11}, \ldots, a_{c1}, \ldots, a_{cn}]$$
$$\mathbf{fx} = [u_1, \ldots, u_c, u_1x_1, \ldots, u_cx_1, \ldots, u_cx_n]^T$$

(iii) Quadratic;

$$\mathbf{a}^T = [a_{10}, \ldots, a_{c0}, a_{11}, \ldots, a_{c1}, \ldots, a_{cn}, \ldots, a_{cnn}]$$
$$\mathbf{fx} = [u_1, \ldots, u_c, u_1x_1, \ldots, u_cx_1, \ldots, u_cx_n, \ldots, u_cx_nx_n]^T$$

(iv) Reduced quadratic;

$$\mathbf{a}^T = [a_{10}, \ldots, a_{c0}, a_{11}, \ldots, a_{c1}, \ldots, a_{cn}, \ldots, a_{cnn}]$$
$$\mathbf{fx} = [u_1, \ldots, u_c, u_1x_1, \ldots, u_cx_1, \ldots, u_cx_n, \ldots, u_cx_n^2]^T$$

For the discriminant function coming in the form of (41), a two-class classifier implements the decision rule expressed by (40). Namely, $\mathbf{x}$ is assigned to ω_1 if the inner product $\mathbf{a}^T\mathbf{fx}$ is greater than zero and to ω_2 otherwise. The relationship $g(\mathbf{x}) = 0$ defines the decision surface that separates the two classes. Otherwise, a multi-class classifier implements the decision rule expressed by (39). Each output node generates a discriminant function corresponding

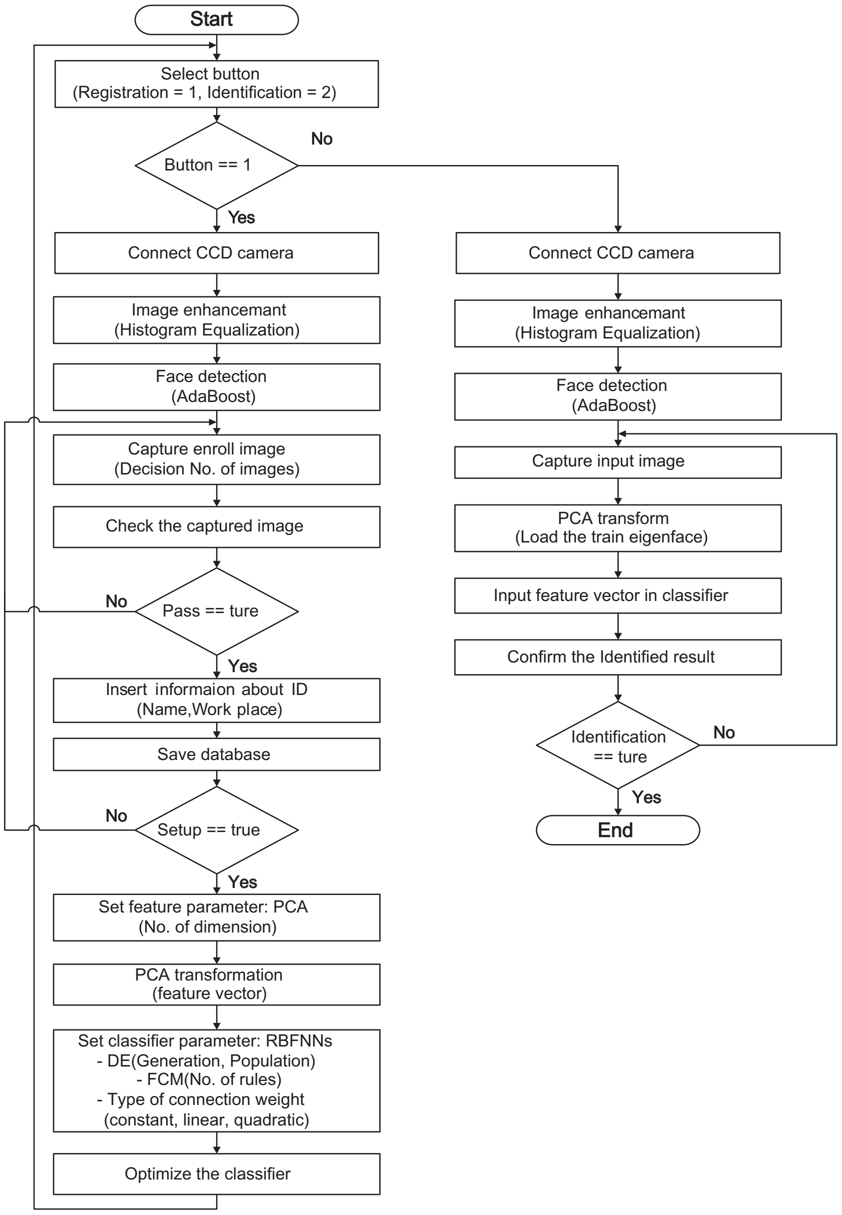

Start
Select button (Registration = 1, Identification = 2)
Button == 1
No
Yes
Connect CCD camera
Image enhancemant (Histogram Equalization)
Face detection (AdaBoost)
Capture enroll image (Decision No. of images)
Check the captured image
Pass == ture
Insert informaion about ID (Name,Work place)
Save database
Setup == true
Set feature parameter: PCA (No. of dimension)
PCA transformation (feature vector)
Set classifier parameter: RBFNNs
- DE(Generation, Population)
- FCM(No. of rules)
- Type of connection weight (constant, linear, quadratic)
Optimize the classifier
Capture input image
PCA transform (Load the train eigenface)
Input feature vector in classifier
Confirm the Identified result
Identification == ture
End

Fig. 13. Processing details realized in the system.

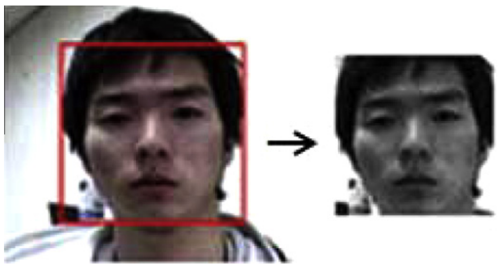


Fig. 14. Process of face region extraction.

to each class. If the i^{th} output is larger than all remaining outputs, pattern $\mathbf{x}$ is assigned to ith class.

4.2. Learning of P-RBF NNs using gradient descent method

In order to estimate the coefficients of the polynomials standing in the conclusion phase of the P-RBF NNs classifier, we consider a gradient descent method. For convenience, we explain the learning process considering the case where the discriminant function is governed by (40). The learning of P-RBF NNs uses gradient descent method augmented by a momentum term. The coefficients are adjusted by taking the negative gradient of the error function E_q expressed as:

$$E_q = \frac{1}{2}(t_q - y_q)^2 \quad q = 1, \ldots, N \tag{42}$$

where N is the size of training data set. t_q is the target output for the qth pattern. $y_q$is the network output when exposed to the qth pattern:

$$y_q = \sum_{i=1}^{c} u_{qi} f_i \tag{43}$$

The learning of the P-RBF NNs involves four cases given a specific form off_i as given by (34)–(37). Those are Constant, Linear, Qua-

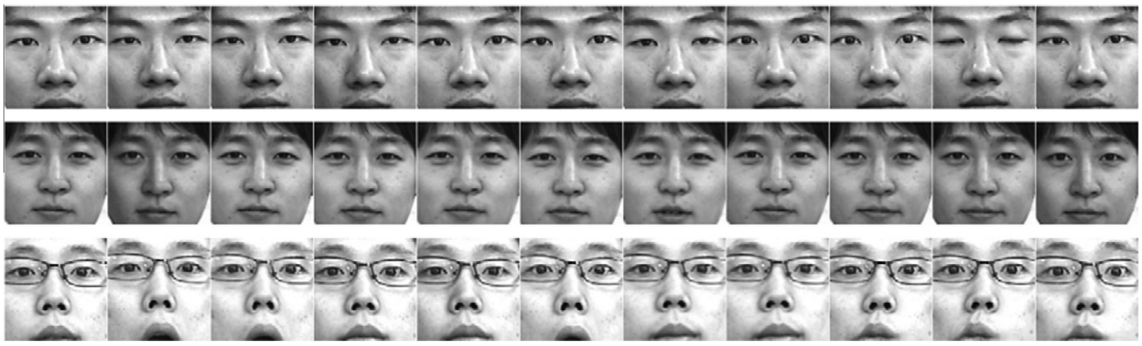


Fig. 15. Sample images used in the system.

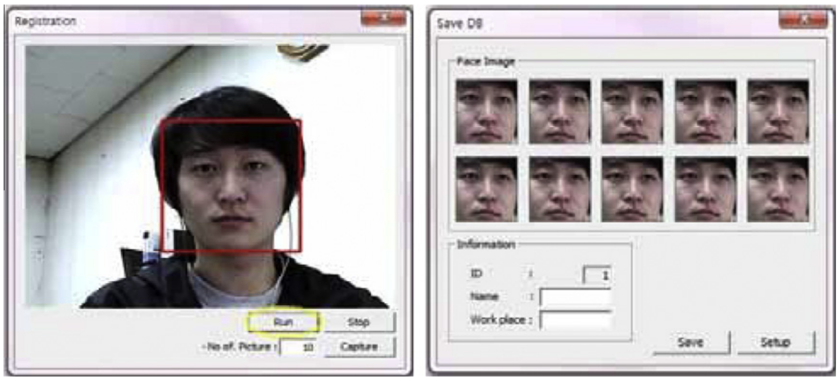
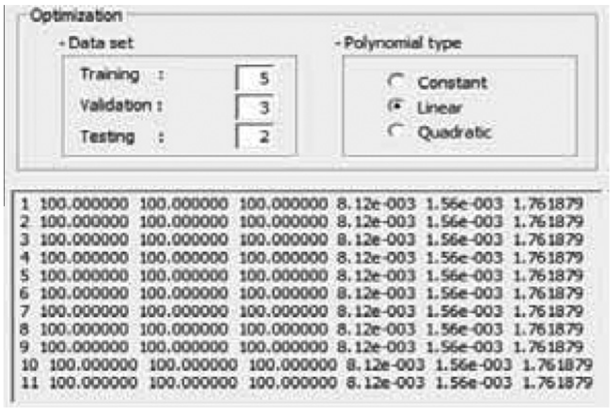


Fig. 16. Registration window.


Optimization

Data set
- Training : 5
- Validation : 3
- Testing : 2

Polynomial type
- Constant
- [✓ Linear]
- Quadratic

```
1 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
2 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
3 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
4 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
5 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
6 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
7 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
8 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
9 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
10 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
11 100.000000 100.000000 100.000000 8.12e-003 1.56e-003 1.761879
```

Fig. 17. Setup module for training.

dratic, Reduced Quadratic type, respectively. The detailed formulas come as follows

If the polynomial is constant then $f_i = a_{i0}$. Here we have:

$$\frac{\partial E_q}{\partial a_{i0}} = \frac{\partial E_q}{\partial y_q} \cdot \frac{\partial y_q}{\partial f_i} \cdot \frac{\partial f_i}{\partial a_{i0}} \tag{44}$$

More specifically the above expressions read as

$$\frac{\partial E_q}{\partial 1pty_q} = -(t_q - y_q) \tag{45}$$

$$\frac{\partial y_q}{\partial f_i} = u_i \tag{46}$$

$$\frac{\partial f_i}{\partial a_{i0}} = 1 \tag{47}$$

Therefore, the error rate $\Delta a_{i0}(l)$ in lth learning iteration can be written

$$\Delta a_{i0}(l) = -\eta \cdot \frac{\partial E_q}{\partial a_{i0}} = \eta(t_q - y_q)u_i \tag{48}$$

In the sequel, when considering momentum, each $a_{i0}(l)$ is updated according to the following rule:

$$a_{i0}(l+1) = a_{i0}(l) + \Delta a_{i0}(l) + \alpha(a_{i0}(l) - a_{i0}(l-1)) \tag{49}$$

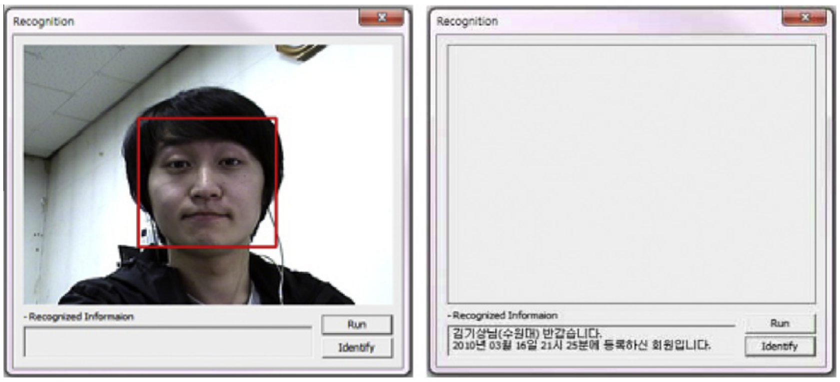
Here, η and α are represent learning rate and momentum coefficient, respectively.

Besides this “constant” polynomial of the conclusion phase explained previously, the learning of polynomial types such as linear, quadratic, and reduced quadratic are also carried out in the same way.

5. Optimization of parameters of the P-RBF NNs with the aid of differential evolution

5.1. Differential evolution (DE)

The differential evolution (DE) algorithm (Dervis & Selcuk, 2004; Storn, 1997) introduced by Storn and Price (1995) is a novel parallel direct search method, which utilizes NP parameter vectors as a population for each generation G. DE can be categorized into a class of floating-point encoded, evolutionary optimization algorithms. Currently, there are several variants of DE. The particular variant used throughout this investigation is the DE/rand/1/bin scheme. This scheme will be discussed here and more detailed descriptions are provided. Since the DE algorithm was originally designed to work with continuous variables, the optimization of continuous problems is discussed first.



김기상님(수원대) 반갑습니다.
2010년 03월 16일 21시 25분에 등록하신 회원입니다.

Fig. 18. Face recognition module.

Table 11
Comparison of results of real-time recognition rate of the two algorithms. (Polynomial type: linear).

Number of rules	PCA (*Case* 1)	PCA-LDA fusion (*Case* 2)
2	91.25% (14/160)	91.88% (13/160)
3	89.38% (17/160)	93.13% (11/160)
4	90.00% (16/160)	93.13% (11/160)
5	88.75% (18/160)	92.50% (12/160)
6	90.00% (16/160)	91.25% (14/160)

The DE algorithm is a population-based algorithm using three operators: crossover, mutation, and selection. Several optimization parameters require tuning. These parameters are put together under the common name control parameters. In fact, there are only three real control parameters of the algorithm, which are differentiation (or mutation) constant F, crossover constant CR, and size of population NP. The rest of the parameters are the dimension of problem D that scales the difficulty of the optimization task; maximum number of generations (iterations) GEN, which may serve as a stopping condition; and the boundary constraints imposed on the variables that limit the feasible area.

5.1.1. Initialization

As with all evolutionary optimization algorithms, DE works with a population of solutions, rather than a single solution. The population *P* at generation *G* contains NP solution vectors called individuals of the population where each vector represents a potential solution for the optimization problem

$$P^{(G)} = X_i^{(G)} = x_i^{(G)}, \quad i = 1, \ldots, NP; \quad j = 1, \ldots, D; \quad G = 1, \ldots, G_{\max} \tag{50}$$

In order to form a starting point for optimum seeking, the population must be initialized. Often there is no specific knowledge available about the location of a global optimum. Typically, we might have knowledge about the boundaries of the problem's variables. In this case, a natural way to initialize the population $P^{(0)}$ (initial population) is to seed it with random values within the given boundary constraints:

$$P^{(0)} = x_{j,i}^{(0)} = x_j^{(L)} + rand_j[0,1] \times \left(x_j^{(U)} - x_j^{(L)}\right) \quad \forall i \in [1, NP]; \quad \forall j \in [1, D] \tag{51}$$

where $rand_j[0,1]$ represents a uniformly distributed random variable assuming values in $[0,1]$.

5.1.2. Mutation

The self-referential population recombination scheme of DE is different from the other evolutionary algorithms. From the first generation onward, the population of the subsequent generation $P^{(G+1)}$ is obtained on the basis of the current population $P^{(G)}$. First a temporary or trial population of candidate vectors for the subsequent generation, $P^{(G+1)} = V^{(G+1)} = v_{j,i}^{(G+1)}$, is generated as follows:

$$v_{j,i}^{(G+1)} = \begin{cases} x_{j,r3}^{(G)} + F \times \left(x_{j,r1}^{(G)} - x_{j,r2}^{(G)}\right), & rand_j[0,1] < CR \vee j = k, \\ x_{ij}^{(G)}, & otherwise, \end{cases} \tag{52}$$

where $i \in [1, \mathrm{NP}]$; $j \in [1, D]$, $r1$, $r2$, $r3 \in [1, \mathrm{NP}]$, randomly selected, except for $r1 \neq r2 \neq r3 \neq i$, $k = (\text{int} \ (\text{rand}_i[0,1] \times D) + 1)$, and $\mathrm{CR} \in [0,1]$, $F \in (0,1]$.

Three randomly chosen indexes, *r*1, *r*2, and *r*3 refer to three randomly chosen vectors of the population. They are mutually different from each other and also different from the running index *i*. New random values for *r*1, *r*2, and *r*3 are assigned for each value of index *i* (for each vector). A new value for the random number rand[0, 1] is assigned for each value of index *j* (for each vector parameter).

5.1.3. Crossover

The index *k* refers to a randomly chosen vector of parameter and it is used to ensure that at least one vector parameter of each individual trial vector $V^{(G+1)}$ differs from its counterpart present in the previous generation $X^{(G)}$. A new random integer value is assigned to *k* for each value of the index *i* (prior to construction of each trial vector). *F* and CR are DE control parameters. Both values remain constant during the search process. Both values as well as the third control parameter, NP (population size), remain constant during the search process. *F* is a real-valued factor in range [0.0, 1.0] that controls the amplification of differential variations. CR is a real-valued crossover factor taking on the values in the range [0.0, 1.0] that controls the probability that a trial vector will be selected form the randomly chosen, mutated vector, $V_{j,i}^{(G+1)}$ instead of from the current vector, $x_{j,i}^{(G)}$ Generally, both *F* and CR affect the convergence rate and the robustness of the search process. Their optimal values are dependent both on the characteristics of the objective function and on the population size, NP. Usually, suitable values for *F*, CR and NP can be found through experimentation after a number of tests using different values. Practical guidelines on how to select control parameters NP, *F* and CR can be found in Storn and Price, 1995 and Storn and Price, 1997.

5.1.4. Selection

The selection scheme of DE differs from the one encountered in other evolutionary algorithms. On the basis of the current population $P^{(G)}$ and the temporary population $P'^{(G+1)}$, the population of the next generation $P^{(G+1)}$ is created as follows:

$$X_i^{(G+1)} = \begin{Bmatrix} V_i^{(G+1)}, & \text{if } \Im\left(V_i^{(G+1)}\right) \leqslant \Im\left(X_i^{(G)}\right) \\ X_i^{(G+1)}, & \text{otherwise} \end{Bmatrix} \tag{53}$$

Thus each individual of the temporary or trial population is compared with its counterpart in the current population. The one with the lower value of cost-function $\Im(X)$ to be minimized will propagate to the population of the next generation. As a result, all the individuals of the next generation are better than their counterparts in the current generation. The interesting point concerning the DE selection scheme is that a trial vector is only compared to one individual vector, not to all the vectors in the current population.

5.1.5. Boundary constraints

It is important to notice that the recombination operation of DE is able to extend the search outside of the initialized range of the search space (refer to (5.2) and (5.3)). It is also worthwhile to notice that sometimes it is a beneficial property in problems with no boundary constraints because it is possible to find the optimum that is located outside of the initialized range. However, in the boundary-constrained problems, it is essential to ensure that parameter values lie inside their allowed ranges after recombination. A simple way to guarantee this is to replace the parameter values that violate boundary constraints with random values generated within the feasible range:

$$u_{j,i}^{(G+1)} = \begin{cases} x_j^{(L)} + rand_j[0,1] \times \left(x_j^{(U)} - x_j^{(L)}\right), & \text{if } u_{j,i}^{(G+1)} < x_j^{(L)} \vee u_{j,i}^{(G+1)} > x_j^{(U)} \\ u_{i,j}^{(G+1)}, & \text{otherwise} \end{cases} \tag{54}$$

where $i \in [1,\text{NP}]$; $j \in [1,D]$.

This is the method used in this work. Another simple but less efficient method is to reproduce the boundary constraint violating values according to relationship (5.3) as many times as necessary to satisfy the boundary constraints. Yet another simple method that allows bounds to be approached asymptotically while minimizing the amount of disruption that results from resetting the bound values is expressed as follows

$$u_{j,i}^{(G+1)} = \begin{cases} \left(x_{j,i}^{(G)} + x_j^{(L)}\right)/2, & \text{if } u_{j,i}^{(G+1)} < x_j^{(L)}, \\ \left(x_{j,i}^{(G)} + x_j^{(U)}\right)/2, & \text{if } u_{j,i}^{(G+1)} < x_j^{(U)}, \\ u_{j,i}^{(G+1)}, & \text{otherwise} \end{cases} \tag{55}$$

5.2. Architecture of vectors in P-RBF NNs

Through the optimization of these parameters such as learning rate, momentum coefficient, fuzzification coefficient and the feature selection by using DE, P-RBF NNs structure exhibits better convergence properties in the generation process of the network from the viewpoint of performance. Fig. 7 shows the structure of parameter vectors used in optimization of the P-RBF NNs.

Individual vectors of the P-RBF NNs include entries that represent optimized learning rate, momentum, fuzzification coefficient, and feature selection. The learning rate and momentum of vectors are applied to optimize the connections (weight) standing in (48) and (49). The fuzzification coefficient changes the shape of the membership functions produced by the Fuzzy C-Means. The values of the membership function depend on the location of the center point and the fuzzification coefficient to be adjusted. In the feature selection part, a feature whose value is greater than 0.45 will be selected.

6. Experimental studies

The complete face recognition system comprises two stages. The First stage requires an extraction of pertinent features from the facial images and the formation of the feature vectors. The second stage involves a classification of facial images based on the derived feature vector obtained in the first stage.

In this experiment, we have used PCA, and PCA-LDA fusion algorithm as a feature domain that uses global data to form the feature vector at the first stage. The designed P-RBF NNs with the proposed learning algorithm has been used as a classifier in the second stage of the face recognition system. Our objective is to quantify the performance of the proposed P-RBF NNs classifier. We compare and analyze the performance of the PCA-LDA fusion algorithm. In the assessment of the performance of the classifier, we report the percentage of correctly classified patterns (classification rate).

6.1. Experimental design

To check the usefulness of the proposed algorithm, the experimental studies are carried out on the Yale face database (available at http://cvc.yale.edu/projects/yalefaces/yalefaces.html) and the AT&T database (available at http://www.uk.research.att.com/facedatabase.html). Table 1 includes a description of these two face image databases.

Each experiment consists of four steps. In the first step, we split data into 50%–30%–20% training, validation, and testing subsets, namely, 80% (50%–30% training and validation set) of the whole pattern are selected randomly for training and the remaining patterns are used for testing purposes. Therefore, in the AT&T database a total of 200 images are used as the training set, 120 are the validation set, and another 80 images are used for testing set while in the Yale database a total of 90 images are used for training, 45 for validation and the rest 30 are used for testing. In the second step PCA (*Case* 1), and PCA-LDA fusion algorithm (*Case* 2) are generated inside the sub-images. In the third step, the classifier is designed and trained. Finally at the fourth step, the performance of the classifier is evaluated. In the assessment of the performance of the classifier, we report the % of correctly classified patterns.

The experiments are completed for a number of scenarios in which we considered a suite of numeric values of some essential parameters. The number of rules varied in-between 2 and 6. We used a reduced quadratic function (37). This reduced version of the quadratic function is helpful in reducing the computing burden when dealing with a large number of combinations of the pairs of the variables. For each combination of the parameters, the experiment was repeated 5 times. The results are reported by presenting the average and standard deviation of the classification rate obtained over these 5 repetitions of the experiment.

The numeric values of the parameters of the DE used in the experiments are covered in Table 2.

6.2. Yale face database

In this experiment, we used PCA (*Case 1*), and PCA-LDA fusion algorithms (*Case 2*) as a feature extraction. The accuracy for each different feature extraction methods are given in Tables 4 and 5. The entries in boldface indicate the corresponding best mean accuracy obtained with features extracted by the corresponding feature extraction methods. Table 6 show the recognition rate of proposed method compared with the average recognition rate of LPP (Locality Preserving Projection) (He & Niyogi, 2003), 2D-PCA (Cai, He, & Han, 2007), 2D-LPP (Yang, Zhang, Frangi, & Yang, 2004), and SR (subspace learning) (Hu, Feng, & Zhou, 2007). The results show that proposed method is superior to the other approaches in all experiments (Table 3).

- It is difficult to use Constant type of polynomial function because of the large number of classes. Therefore here we consider the Linear, and reduced quadratic types.
- The standard deviation of testing data is larger than that of the training data and the L-RBF NNs model produces the best performance (recognition rate for testing data is 95.57 ± 2.49%) when the number of rules equals to 2.
- Fig. 10(a) depicts shows a comparative analysis results of between the fusion method, and PCA. It is clear that the fusion method, and PCA better performance can be obtained with the increasing number of rules.
- When the number of rules increases, the fusion method leads to the better performance in comparison with PCA-based method

The table demonstrates that in all experiments the proposed methods perform better than the other approaches reported in the literature (see Fig. 11).

6.3. AT&T database

The experiments were performed exactly in the same way as for the Yale face database (see Tables 7–10).

Several main observations are to be made here:

- The standard deviation on the testing data is larger than that for the training data.
- The linear type of polynomial function produces better performance than the reduced quadratic.
- When the number of rules increases, the performance becomes worse on testing data. The L-RBF NNs model obtains the best performance (with the recognition rate for the testing data equal to 95.25 ± 2.36%) when the number of rules equals to 4.
- When the number of rules increases, the fusion method leads to the better performance in comparison with the PCA-based method.
- Experimental results showed that the fusion method has improved when compared with PCA in terms of P-RBF NNs models.

6.4. Application to real-time face recognition system

In this section, we show an application of the real-time face recognition system. The overall system is divided into two basic modules, namely the preprocessing and recognition part. In the preprocessing part, two dimensional gray images of face are obtained by using AdaBoost. In the sequel histogram equalization is used to improve the quality of image. In the recognition part, the optimized P-RBF NNs are used. This system has been designed for the facial recognition security system. The overall architecture of the system is presented in Fig. 12 while Fig. 13 provides a more detailed view at the processing realized there.

In this paper, we use the AdaBoost algorithm with the Haar-like feature for face detection. AdaBoost (Adaptive Boosting), which is a widely used machine learning algorithm (Freund & schapire, 1995) The AdaBoost method is combined with the Haar-like feature to improve the performance of the system. The process of face region extraction is shown in Fig. 14.

The system was implemented by Microsoft Visual C++ 2008. It accepts input images through a CCD camera in a continuous way. The captured image are then cropped by the face detection module; one saves only the gray scale facial information in the BMP format of 200×200 matrix size. The images are saved in sequence of their occurrence time. Fig. 15 shows the sample images captured by the system.

In the database there are 160 samples, 10 images for each student in different position and with different emotional patterns. The features of facial images were extracted by PCA method. We split data into 50%–30%–20% training, validation, and testing subsets, namely, 80% (50%–30% training and validation set) of the entire set of patterns are selected randomly for training while the remaining patterns are used for testing purposes. As noted, the parameters such as the learning rate, momentum coefficient, and fuzzification coefficient are optimized by means of the DE method. various windows of the proposed system are shown as well. Fig. 16 shows the registration window for collecting the dataset. In Fig. 17, the setup module for the training is shown while Fig. 18 presents the face recognition module. A way of displaying values of the real-time recognition rate in presented in Table 11. In this experiment, we used PCA (*Case* 1), and PCA-LDA fusion algorithms (*Case* 2) as the mechanisms of feature extraction.

The values of the performance index of each experiment are as the classification rates. The recognition rate of all the experiments is evaluated as the number of recognized faces for 10 times per candidate. Table 11 shows the performance of the proposed PCA (*Case* 1) and the PCA-LDA fusion methods (*Case* 2) based on the p-RBF neural networks classifier. The *Case* 2 shows the highest recognition rates. The performance produced by the PCA-LDA fusion method is higher than the one obtained in case of the PCA method.

7. Conclusions

In this paper, we have proposed the face recognition technique for image feature extraction and recognition. Also we elaborated on the implementation of real-time face recognition system. The PCA-LDA fusion algorithm used in the preprocessing part exhibits many advantages over the conventional PCA (Eigenfaces). Since the method is based on the image matrix, it is simpler and more straightforward to use for feature extraction and better than PCA in terms of recognition rate. In the recognition part, the proposed P-RBF NNs exhibit some unique and useful characteristics. The P-RBF NNs involve a partition module formed by the FCM clustering and used here as an activation function of the neurons located in the hidden layer. The proposed P-RBF NNs are expressed as a collection of "if–then" fuzzy rules. The resulting model is endowed with polynomial weights (functional links). Given this, the network is capable of generating more complex nonlinear discriminant functions. The estimation of some parameters of the P-RBF NNs such as the learning rate, momentum coefficient and fuzzification coefficient by means of differential evolution (DE) appeared to be an important and effective design process. Owing to this form of optimization, we can produce a classifier that can effectively cope with the nonlinearity present in the underlying classification problem. The experiments quantified the classification capabilities of the P-RBF NNs.

The proposed P-RBF NNs could be of interest as computationally effective constructs for handling high-dimensional pattern classification problems. To handle these tasks, our plan is to consider feature aspects including information data preprocessing by means of evolutionary algorithms including DE.

Acknowledgements

This work was supported by National Research Foundation of Korea Grant funded by the Korean Government (NRF-2012-003568) and this work was partly supported by the GRRC program of Gyeonggi province [GRRC Suwon 2012-B2, Center for U-city Security & Surveillance Technology].

References

Aiver, A., Pyun, K., Huang, Y. Z., O'Brien, D. B., & Gray, R. M. (2005). Lloyd clustering of Gauss mixture models for image compression and classification. *Signal Processing: Image Communication, 20*(5), 459–485.

Balasubramanian, M., Palanivel, S., & Ramalingam, V. (2009). Real time face and mouth recognition using radial basis function neural networks. *Expert Systems with Applications, 36*, 6879–6888.

Belhumeur, P., Hespanha, J., & Kriegman, D. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*(7), 711–720.

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms*. Plenum Press.

Cai, D., He, X., & Han, J. (2007). Spectral regression for efficient regularized subspace learning. In *IEEE international conference on computer vision (ICCV), Rio de Janeiro, Brazil*.

Cho, K. B., & Wang, B. H. (1996). Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction. *Fuzzy Sets and Systems, 83*, 325–339.

Dervis, K., & Selcuk, O. (2004). A simple and global optimization algorithm for engineering problems: Differential evolution algorithm. *Turkish Journal of Electrical Engineering, 12*, 53–60.

Er, M. J., Wu, S. Q., Lu, J. W., & Toh, H. L. (2002). Face recognition with radial basis function (RBF) neural networks. *IEEE Transactions on Neural Networks, 13*(3), 697–710.

Freund, Y., & Schapire, R. E., (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory: Eurocolt'95* (pp. 23–37).

Gumus, E., Kilic, N., Sertbas, A., & Ucan, O. N. (2010). Evaluation of face recognition technique using PCA, wavelets and SVM. *Expert Systems with Applications, 37*, 6404–6408.

Haykin, S. (1999). Neural Networks: a comprehensive foundation, (2nd ed.)., Prentice-Hall.

He, Z., & Niyogi, P. (2003). Locality preserving projections. In *NIPS* (Vol. 16).

Hu, D., Feng, G., & Zhou, Z. (2007). Two-dimensional locality preserving projections (2DLPP) with its application to palmprint recognition. *Pattern Recognition, 40*(1), 339–342.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transaction in Neural on Networks, 8*, 98–113.

Lee, C., & Landgrebe, D. A. (1997). Decision boundary feature extraction for neural networks. *IEEE Transactions on Neural Networks, 8*, 75–83.

Lin, S.-H., Kung, S.-Y., & Lin, L.-J. (1997). Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural on Networks, 8*, 114–132.

Lippman, R. P. (1981). An introduction to computing with neural nets. *IEEE ASSP Magazine, 4*(2), 4–22.

Mali, K., & Mitra, S. (2005). Symbolic classification, clustering and function network. *Fuzzy Sets and Systems, 152*, 553–564.

Oh, S.-K., Pderyz, W., & Park, B.-J. (2004). Self-organization neurofuzzy networks in modeling software data. *Fuzzy Sets and Systems, 145*, 165–181.

Park, B.-J., Oh, S.-K., & Kim, H.-K. (2008). Design of polynomial neural network classifier for pattern classification with two classes. *Journal of Electrical Engineering & Technology, 3*(1), 108–114.

Patrikar, A., & Provence, J. (1992). Pattern classification using polynomial networks. *Electronics Letters, 28*(12), 1109–1110.

Sirovich, L., & Kirby, M., (1987). Low-Dimensional Procedure for Characterization of Human faces. *J. Optical Soc. am., 4*, 519–524.

Song, H. H., & Lee, S. W. (1998). A self-organizing neural tree for large-set pattern classification. *IEEE Transactions on Neural Networks, 9*, 369–380.

Staiano, A., Tarliaferri, R., & Pedrycz, W. (2006). Improving RBF networks performance in regression tasks by means of supervised fuzzy clustering. *Neurocomputing, 69*, 1570–1581.

Storn, R. (1997). Differential evolution, a simple and efficient heuristic strategy for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359.

Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience, 3*(1), 71–86.

Yang, J., Zhang, D., Frangi, A. F., & Yang, J. (2004). Two-dimensional PCA: A new approach to appearance- based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine intelligence, 26*(1), 131–137.

Zhao, W., Chellappa, R., & Krishnaswamy, A. (1998). Discriminant analysis of principal components for face recognition. In *Proceedings of the third international conference on automatic face and gesture recognition* (pp. 336–341).

Zhou, W. (1999). Verification of the nonparametric characteristics of back propagation neural networks for image classification. *IEEE Transactions on Geoscience and Remote Sensing, 32*(2), 771–779.