Bagging Evolutionary Feature Extraction Algorithm for Classification*

Tianwen Zhao
Department of Computer Science
Shanghai Jiao Tong University
Shanghai, 200030, China
ztw505xyz@sjtu.edu.cn

Qijun Zhao
Biometrics Research Center
Department of Computing
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
csqjzhao@comp.polyu.edu.hk

Hongtao Lu
Department of Computer Science
Shanghai Jiao Tong University
Shanghai, 200030, China
lu-ht@cs.sjtu.edu.cn

David Zhang
Biometrics Research Center
Department of Computing
The Hong Kong Polytechnic University
Kowloon, Hong Kong
csdzhang@comp.polyu.edu.hk

Abstract

Feature extraction is significant for pattern analysis and classification. Those based on genetic algorithms are promising owing to their potential parallelizability and possible applications in large scale and high dimensional data classification. Most recently, Zhao et al. presented a direct evolutionary feature extraction algorithm(DEFE) which can reduce the space complexity and improve the efficiency, thus overcoming the limitations of many genetic algorithm based feature extraction algorithms(EFE). However, DEFE does not consider the outlier problem which could deteriorate the classification performance, especially when the training sample set is small. Moreover, when there are many classes, the null space of within-class scatter matrix(S_w) becomes small, resulting in poor discrimination performance in that space. In this paper, we propose a bagging evolutionary feature extraction algorithm(BEFE) incorporating bagging into a revised DEFE algorithm to improve the DEFE's performance in cases of small training sets and large number of classes. The proposed algorithm has been applied to face recognition and testified using the Yale and ORL face databases.

1. Introduction

Feature extraction, as a significant step of pattern classification, acquires a new smaller feature subset to represent the data via transforming the original features. According to whether the transformation can be expressed in matrix form, feature extraction methods can be categorized into linear and nonlinear ones. Compared with nonlinear feature extraction, linear feature extraction, which is the main concern of this paper, is simple to use and also very efficient for many cases.

Feature extraction is essentially a kind of optimization problems. As for linear feature extraction, its task is to seek for *m* projection basis vectors $w_1, w_2, \cdots, w_m \in R^n$, i.e. the column vectors in the transform matrix *W*, such that the resulting transformed samples are most distinguishable. Therefore, feature extraction can acquire its solutions via such optimization techniques as Genetic Algorithms (GA) [2]. These GA based algorithms are called evolutionary feature extraction algorithms (EFE). Compared with other feature extraction algorithms, EFE algorithms have the advantage of potential parallelizability and are thus expected to be more applicable for large-scale and high-dimensional data.

Many researchers [4][6][5][3][9][11][8] have already explored GAs for feature extraction and proposed many GA based feature extraction algorithms, but these algorithms still have some limitations. For example, the space complexity is high, making them unsuitable for high dimensional and large scale data; the search is not well guided or constrained, making them have low search efficiency. Recently, Zhao et al. [10] proposed a direct EFE algorithm(DEFE) which has lower space complexity and higher search efficiency than their counterparts of GA based feature extraction algorithms. Another side product of this al-

*This work is supported by NSFC under project No.60573033.

Third International Conference on Natural Computation (ICNC 2007)
0-7695-2875-9/07 $25.00 © 2007 IEEE

gorithm is that it provides a simple but effective way to investigate the discriminability of different subspaces of the original data space. However, in their experiments, they used the leave-one-image-out evaluation strategy, in which case the training set has relatively large size. When the training set is small, the DEFE algorithm, as a subspace idea based method, could suffer severely from the outlier problem. Besides, in their paper, they searched for the optimal projection basis in the null space of S_w. When there are many classes, this space becomes very small and thus deteriorates the final classification performance.

In this paper, to overcome the limitations of the DEFE algorithm mentioned above, we first revise the current DEFE algorithm using Whitened Principal Component Analysis (WPCA) and weighted fitness; and then we incorporate the bagging method into the revised DEFE, presenting a novel bagging evolutionary feature extraction (BEFE) algorithm. The rest of this paper is organized as follows. In section 2, we describe the revised DEFE algorithm. In section 3, we present our proposed BEFE algorithm in detail. We then show our face recognition experiments in section 4 and conclude this paper in the last section.

2 The Revised Direct Evolutionary Feature Extraction Algorithm

In this section, we will present the revised direct evolutionary feature extraction algorithm in detail. In this revised version, data are first preprocessed using WPCA and a weight is introduced into the fitness function. For the simplicity sake, however, we will still use DEFE to denote the revised direct evolutionary feature extraction algorithm.

2.1 Preprocessing Data: Centering and Whitening

All the data are first preprocessed by centering, i.e. the total mean is subtracted from them:

$$\bar{x}_i = x_i - M, \tag{1}$$

for all $i \in 1, 2, \cdots, N$(N is the number of samples).

DEFE further whitens the centered data to normalize their variance to unity. The whitening transformation matrix is

$$W_{WPCA} = [\frac{\alpha_1}{\sqrt{\lambda_1}} \frac{\alpha_2}{\sqrt{\lambda_2}} \cdots \frac{\alpha_{N-1}}{\sqrt{\lambda_{N-1}}}]. \tag{2}$$

Here, $\{\lambda_i\}$ and $\{\alpha_i\}$, for all $i \in 1, 2, \cdots, N-1$, are the eigenvalues of the covariance matrix and the corresponding eigenvectors respectively.
Let $\bar{X}$ and $\tilde{X}$ be the centered and whitened data, then

$$\tilde{X} = W_{WPCA}^T \bar{X}. \tag{3}$$

2.2 Calculating the Constrained Search Space

According to the Fisher criterion, the most discriminative directions are most probably to lie in the subspaces generated from S_w and S_b.

DEFE calculates the eigenvectors of S_w with positive eigenvalues via an $N \times N$ matrix $S_w^{'}$. Let

$$H_w = [x_1 x_2 \cdots x_N] \in R^{n \times N}, \tag{4}$$

then

$$S_w = \frac{1}{N} H_w H_w^T. \tag{5}$$

$$S_w^{'} = \frac{1}{N} H_w^T H_w, \tag{6}$$

and assume $(\lambda, \alpha_i^{'})$ are an eigenvalue and the corresponding eigenvector of $S_w^{'}$, i.e. We can prove that $(\lambda, H_w \alpha_i^{'})$ are eigenvalue and eigenvector of S_w. A basis of range(S_w) is then given by (L:the number of classes)

$$B_{range}(S_w) = \{\alpha_i = H_w \alpha_i^{'} | i = 1, 2, \cdots, N - L\}. \tag{7}$$

The basis of range(S_b) can be calculated in a similar way. And based on the basis of range(S_w), it is easy to get the basis of null(S_w) through calculating the orthogonal complement space of range(S_w).

2.3 Searching: An Evolutionary Approach

2.3.1 Encoding Individuals

First, DEFE generates one vector via linearly combining the basis of the search space. Suppose the search space is R^n and let $\{e_i \in R^n | i = 1, 2, \cdots, n\}$ be a basis of it, $\{a_i \in R | i = 1, 2, \cdots, n\}$ be the linear combination coefficients. Then

$$v = \sum_{i=1}^{n} a_i e_i. \tag{8}$$

Second, DEFE calculates a basis of the orthogonal complement space of V=span{v}, the space expanded by v, in R_n. Let $\{u_i \in R^n | i = 1, 2, \cdots, n-1\}$ be the basis, and $U = span\{u_1, u_2, \cdots, u_{n-1}\}$, then

$$R^n = V \bigoplus U, \tag{9}$$

where $' \bigoplus '$ represents the direct sum of vector spaces, and

$$U = V^{\perp}, \tag{10}$$

where $' \perp '$ denotes the orthogonal complement space. Finally, we randomly choose part of this basis as the projection basis vectors.

According to the above method of generating projection basis vectors, the information encoded in an individual includes the n combination coefficients and (n-1) selection bits. Each coefficient is represented by 11 bits with the leftmost bit denoting its sign ('0' means negative and '1' positive) and the remaining 10 bits giving its value as a binary decimal. Figure. 1 shows such individual, in which the selection bits $b_1, b_2, \cdots, b_{n-1}$, taking the value of '0' or '1' indicate whether the corresponding basis vector is chosen as a projection basis vector. The individual under such definition has (12n-1) bits.

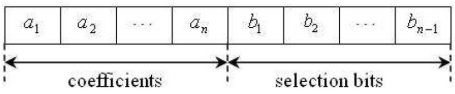


Figure 1. The individual defined in the evolutionary feature extraction. Each coefficient is represented with 11 bits.

2.3.2 Evaluating Individuals

From the perspective of pattern recognition, almost all existing GA based feature extraction algorithms evaluate individuals using the classification accuracy in the obtained feature space on the training samples or a subset of them. However, after being preprocessed using WPCA, the classification accuracy on the training samples is always almost 100%. In [11], Zheng *et al.* also find this when they use PCA to process the training data, and they then just ignore its role in evaluating individuals. Different from their method, we keep this classification accuracy term in the fitness function, but based on a validation set, instead of the training set (the current DEFE algorithm uses a subset of training set, called the boundary set). Specifically, we randomly choose from each cluster N_{va} samples to create a validation set Ω_{va} and the remaining N_{tr}=$(N - L \times N_{va})$ samples are used as the training set Ω_{tr}. Assume $N^c_{va}(D)$ samples in the validation set are correctly classified in the feature space learned from Ω_{tr} by an individual *D*, the classification accuracy term for this individual is then defined as

$$\zeta_a(D) = N^c_{va}(D)/N_{va}. \tag{11}$$

From the machine learning perspective, the generalization ability is an important index of machine learning systems. In previous methods, the between-class scatter is widely used in fitness functions. However, according to the Fisher criterion, it is better to simultaneously minimize the within-class scatter while maximizing the between-class scatter. Thus, we use the following between-class and within-class scatter distances of samples in the feature space,

$$d_b(D) = \frac{1}{N}\sum_{j=1}^{L} N_j (M_j - M)^T (M_j - M) \tag{12}$$

and

$$d_w(D) = \frac{1}{L}\sum_{j=1}^{L} \frac{1}{N_j}\sum_{i \in I_j} (y_i - M_j)^T (y_i - M_j), \tag{13}$$

to measure the generalization ability as

$$\zeta_g(D) = d_b(D) - d_w(D). \tag{14}$$

Here, the means, *M* and $M_j, j = 1, 2, \cdots, L$, are calculated based on $\{y_i | i = 1, 2, \cdots, N\}$ in the feature space.

Summarizing, we define the fitness function as the weighted summation of the above two terms as follows

$$\zeta(D) = \pi_a \zeta_a(D) + (1 - \pi_a)\zeta_g(D), \tag{15}$$

where $\pi_a \in [0, 1]$ is the weight. The accuracy term ζ_a in this fitness function lies in the interval [0, 1]. Thus, it is reasonable to make the value of the second generalization term ζ_g to be of a similar order. This demonstrates the essential of preprocessing data by centering and whitening as discussed in the beginning of this section.

2.3.3 Generating New Individuals

DEFE uses three genetic operators: selection, crossover, and mutation. The selection is based on the relative fitness of individuals. Specifically, the proportion of the fitness of an individual to the total fitness of the population determines how many times the individual will be selected as parent individuals. After evaluating all individuals in the current population, DEFE selects (S-1) pairs of parent individuals from them, where S is the size of the GA population. (S-1) new individuals generated from these parent individuals together with the individual with the highest fitness in current generation form the population of the next generation.

The crossover operator is conducted under a given probability. If two parent individuals are not subjected to crossover, the one having higher fitness will be chosen into the next generation. Otherwise, two crossover points are randomly chosen, one of which is within the coefficient bits and the other is within the selection bits. These two points divide both parent individuals into three parts, and the second part is then exchanged between them to form two new individuals, one of which is randomly chosen as an individual in the next generation.

At last, each bit in the (S-1) new individuals is subjected to mutation from '0' to '1' or reversely under a specific probability. After applying all the three genetic operators, we have a new population for the next GA iteration.

2.3.4 Imposing Constraints on Searching

DEFE is to construct vectors by linearly combining the basis of the constrained search space, instead of the original space. Take $null(S_w)$, the null space of S_w, as an example. Suppose we want to constrain the GA to search in $null(S_w)$. Let $\{\alpha_i \in R^n | i = 1, 2, \cdots, m\}$ be the eigenvectors of S_w associated with zero eigenvalues. They form a basis of $null(S_w)$. After obtaining a vector v via linearly combining the above basis, we have to calculate the basis of the orthogonal complement space of V = span{v} in the constrained search space $null(S_w)$, but not the original space R^n. For this purpose, DEFE first calculates the isomorphic space of V in R^m, denoted by $\hat{V} = span\{P^T v\}$, where $P = [\alpha_1 \alpha_2 \cdots \alpha_m]$ is an isomorphic mapping. Then calculate a basis of the orthogonal complement space of $\hat{V}$ in R^m. Let $\{\hat{\beta}_i \in R^m | i = 1, 2, \cdots, m-1\}$ be the obtained basis. Finally, map this basis back into $null(S_w)$ through $\{\beta_i = P\hat{\beta}_i \in R^n | i = 1, 2, \cdots, m-1\}$.

3 Bagging Evolutionary Feature Extraction

As we said before, the above DEFE algorithm could suffer from the outlier problem especially when the training set is small. Moreover, when there are many classes, the $null\{S_w\}$ space becomes small, resulting in poor discrimination performance in that space. Wang and Tang studied this in [7] and proposed to address the problem using two random sampling techniques, random subspace and bagging. To improve the performance of the evolutionary feature extraction framework on large-scale data sets, we also use the bagging method and incorporate it with the evolutionary feature extraction. In this section, we will introduce the bagging evolutionary feature extraction in detail.

Bagging (acronym for Bootstrap Aggregating), proposed by Breiman [1], uses re-sampling to generate several random subsets (called random bootstrap replicates) from the whole training set. From each replicate, one classifier is constructed. The results from these classifiers are integrated using some fusion scheme to give the final result. Since these classifiers are trained from relatively small bootstrap replicates, the over-fitting problem for them can be somewhat alleviated. Besides, the stability of the overall classifier system can be also improved by integration of several (weak) classifiers.

Like Wang and Tangs method, we randomly choose some classes from all the classes in the training set. The training samples belonging to these classes compose a bootstrap replicate. Usually, the unchosen samples become useless in the learning process. Instead, we do not overlook these data, but rather use them for validation and calculate the classification accuracy term of the fitness function based on them. Below are the primary steps of our proposed bagging evolutionary feature extraction:

(1) Preprocess the data using centering and whitening.

(2) Randomly choose some classes, say $\hat{L}$ classes, from all the L classes in the training set. The samples belonging to the $\hat{L}$ classes compose a bootstrap replicate used for training, and those belonging to the other $(L - \hat{L})$ classes are used for validation. Totally, K replicates are created (different replicates could have different classes).

(3) Run the DEFE on each replicate to learn a feature space. In all, K feature spaces are obtained.

(4) Classify a new sample using a classifier in the K feature spaces respectively. The resulting K results are combined by a fusion scheme, giving the final result.

There are two basic problems with the bagging evolutionary feature extraction: how to do validation and classification, and how to fuse the results from different replicates? Before closing this section, we will present our solutions to them.

3.1 Validation and Classification

As shown above, a training replicate is created from the chosen $\hat{L}$ classes. Based on this training replicate, an individual in the DEFE population generates a candidate projection basis of feature space. All the samples in the training replicate are projected into this feature space along the projection basis. The generalization term in the fitness function is then calculated from these projections. To get the value of the classification accuracy term, we again randomly choose some samples from all the samples of each class in the $(L - \hat{L})$ validation classes to form the validation set. We then project the remaining samples in these classes to the feature space, and calculate the mean as the prototype for each validation class according to the projections. Finally, the chosen samples are classified based on these prototypes using a classifier. The obtained classification rate is used as the value of the classification accuracy term in the fitness function.

After the DEFE runs on all the replicates, we get K feature spaces as well as one projection basis for each of them. For each feature space, all the training samples (including the samples in training replicates and validation classes) are projected into the feature space and the means of all classes are calculated as the prototypes of them. To classify a new sample, we first classify it in each of the K feature space respectively based on the class prototypes in that space and then combine the K results to give the final decision using a fusion scheme, which will be introduced in the next part.

3.2 Fusion Schemes

A number of fusion schemes have been proposed in the literature of multiple classifiers and information fusion. In the present paper, we only focus on Majority Voting (MV) for its intuitiveness and simplicity.

Let $\{M_j^k \in R^{l_k} | j = 1, 2, \cdots, L; k = 1, 2, \cdots, K\}$ be the prototype of class j in the k_{th} feature space, whose dimension is l_k. Denote the preprocessed sample as x_t. It is then projected into each of the K feature spaces, giving y_t^k in the k_{th} feature space, and classified in these feature spaces respectively. After that, the following fusion scheme is employed to combine the results obtained in the K feature spaces.

The majority voting (MV) is one of the simplest and most popular classifier fusion schemes. Take the Nearest Mean Classifier (NMC) and the k_{th} feature space as an example. The NMC assigns x_t to the class $c^k \in 1, 2, \cdots, L$ such that

$$\|y_t^k - M_{c^k}^k\| = \min_{j \in \{1,2,\cdots,L\}} \|y_t^k - M_j^k\|. \quad (16)$$

In other words, it votes for the class whose prototype is closest to y_t^k. After classifying x_t in all the K feature spaces, we get K results $\{c^k | k = 1, 2, \cdots, K\}$. Let Votes(i) be the number of votes obtained by the class i, i.e.

$$Votes(i) = \#\{c^k = i | k = 1, 2, \cdots, K\}, \quad (17)$$

where '#' denotes the cardinality of a set. The final class label of x_t is then determined to be c if

$$Votes(c) = \max_{i \in \{1,2,\cdots,L\}} Votes(i). \quad (18)$$

4 Face Recognition Experiments

The proposed algorithms have been assessed using face recognition experiments. In this section, we will present our comprehensive experiments.

4.1 The Face Databases

Two popular face databases, the Yale database and the ORL database, were used. The images in Yale database were cropped to retain only the face area and calibrated according to the eye centers and the average face shape, while those in ORL database were used directly. See the second to fourth column of Figure 2[1] for details.

4.2 Parameter Settings

Several parameters are involved. As for these related to the genetic algorithm, we set the probability of crossover to 0.8, the probability of mutation to 0.01, the size of population to 100, and the number of generations to 200, based on a series of experiments. The weight of the classification accuracy term in the fitness function and the number of bagging replicates are evaluated for various choices. The results will be presented in the next sub-section.

All the sample images in each database were first randomly divided into two parts: the training set and the test set. In the experiments with EP (Evolutionary Pursuit)[3], 6 and 5 images were chosen from each subject for training on Yale and ORL databases respectively and the remaining ones were used for test. In the experiments with DEFE, another validation set is further divided from the whole training set. For all databases, we randomly chose one sample out from the training samples of each subject to create the validation set. The case was a little bit different for the experiments with BEFE, where the division of training samples is on the class level (each subject is a class). Specifically, a subset of classes were randomly chosen. The samples belonging to these classes were used for training whereas those belonging to the other classes composed the validation set. From each class in the validation set, some samples were randomly chosen to calculate the prototype of that class and the remaining ones were used for evaluation. The last three columns in Figure 2[1] summarize these settings.

DB	#Sub	Image Size	#Image/Sub	#Train	#Valid	#Test
Yale	15	121×168	11	5	1	5
ORL	40	92×112	10	4	1	5

Figure 2. General Information Of The Databases Used.

Totally, we created 5 pairs of training and test sets, ran algorithms over them respectively, and calculated the recognition rate for each algorithm.

4.3 Results

Then, we evaluate the classification performance of our BEFE and compare it with the result of DEFE, EP.

We first evaluate the algorithms on the Yale database. In the experiments, we use WPCA to reduce the data into an *(N-1)*, where *N* is the number of training samples, dimensional space before applying the EP. As for DEFE, we test its performance using $0.0, 0.1, 0.2, \cdots, 0.9, 1.0$ as the weight of fitness function respectively and take the best result among them. And in BEFE, the number of bagging

[1] From the first column to the last column: the name of the database, the number of subjects, the size of images, the number of images per subject, the number of training samples per subject, the number of validation samples per subject, and the number of test samples per subject.

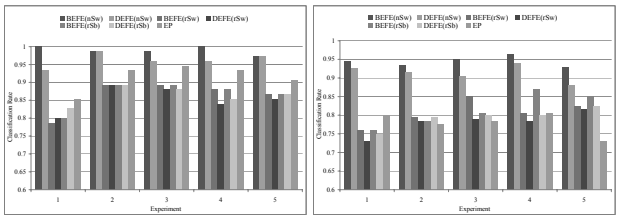
Exp	EP	DEFE			BEFE		
		nS_w	rS_w	rS_b	nS_w	rS_w	rS_b
1	85%	93%	80%	83%	100%	79%	80%
2	93%	99%	89%	89%	99%	89%	89%
3	95%	96%	88%	88%	99%	89%	89%
4	93%	96%	84%	85%	100%	88%	88%
5	91%	97%	85%	87%	97%	87%	87%

(a)

Exp	EP	DEFE			BEFE		
		nS_w	rS_w	rS_b	nS_w	rS_w	rS_b
1	80%	93%	73%	75%	95%	76%	76%
2	78%	92%	79%	80%	94%	80%	79%
3	79%	91%	79%	80%	95%	85%	81%
4	81%	94%	79%	80%	97%	81%	87%
5	73%	88%	82%	83%	93%	83%	85%

(b)

Figure 3. Recognition Rates on the (a)Yale and (b)ORL Databases



(a) (b)

Figure 4. [2]Results on the (a)Yale and (b)ORL Databases

replicates is set to 3, 5, 7 and 9 respectively, also taking the optimal result. One thing should be mentioned that we conduct the tests on three spaces: null space of S_w, range space of S_w, and range space of S_b. Figure 3(a) and Figure 4(a) [2] show the resulting recognition rates of the algorithms on the Yale database. The results illustrate that in Experiment 1, the recognition rates of the $BEFE(rS_w), BEFE(rS_b)$ are a little lower than those of $DEFE(rS_w), DEFE(rS_b)$, but as for the other recognition rates, BEFE is better than DEFE. Especially in the null space of S_w, all the recognition rates of $BEFE(nS_w)$ are much higher than those of its competitive counterparts, including EP, some even achieving the best recognition rate of 100%.

We then compare the algorithms on the ORL database, a database having more classes. The parameter settings are the same as Yale database. Figure 3(b) and Figure 4(b)[2] display the results. Just like the Yale database, in most cases, BEFE overwhelms DEFE and EP. Here, one thing should be pointed out that ORL database has 40 classes, much more than the Yale database, thus resulting in the lower classification rates in $DEFE(nS_w)$ of the ORL database than that of the Yale database as we mentioned in Section 2. Thanks to BEFE, the performance of $BEFE(nS_w)$ in the two databases are approximate.

According to above results, it is obvious that previous DEFE algorithm works poorer with small training set. In most situations, the proposed $BEFE(rS_w)$ and $BEFE(rS_b)$ perform better than those of DEFE. Moreover, the $BEFE(nS_w)$ obtains the best recognition rates in almost all the tests and obviously it's a stable method. All these results testify the feasibility and advantages of the BEFE algorithm proposed in this paper.

5 Conclusions

In this paper, we propose a new evolutionary approach of feature extraction for classification, namely bagging evolutionary feature extraction (BEFE). The previous DEFE method provides an effective alternative way for classical feature extraction methods like PCA and LDA widely used in such real-world applications as face recognition and is more applicable for high-dimensional data because of its low space complexity. But when the training set is small and there are many classes, it could suffer from the outlier problem and will result in poor discrimination performance. Our method can improve the performance of DEFE algorithm. This has been well demonstrated by our extensive face recognition experiments.

[2]For each experiment of the two databases, from the first column to the last column: BEFE(nS_w), DEFE(nS_w), BEFE(rS_w), DEFE(rS_w) , BEFE(rS_b), DEFE(rS_b) and EP.

References

[1] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[2] D. Goldberg. *Genetic Algorithm in Search, Optimization, and Machine Learning*. Adison-Wesley, 1989.

[3] C. Liu and H. Wechsler. Evolutionary pursuit and its application to face recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(6):570–582, 2000.

[4] M. Pei et al. Genetic algorithms for classification and feature extraction. Presented at the Annual Meeting of the Classification Society of North America, 1995.

[5] M. Raymer et al. Dimensionality reduction using genetic algorithms. *IEEE Transaction on Evolutionary Computation*, 4(2):164–171, 2000.

[6] H. Vafaie and K. De Jong. Feature space transformation using genetic algorithms. *IEEE Intelligent Systems and Their Applications*, 13(2):57–65, 1998.

[7] X. Wang and X. Tang. Random sampling for subspace face recognition. *International Journal of Computer Vision*, 70(1):91–104, 2006.

[8] Q. Zhao et al. A fast evolutionary pursuit algorithm based on linearly combining vectors. *Pattern Recognition*, 39:310–312, 2006.

[9] Q. Zhao and H. Lu. GA-driven LDA in KPCA space for facial expression recognition. In *Lecture Notes on Computer Science 3611*, pages 28–36, 2005.

[10] Q. Zhao, D. Zhang, and H. Lu. A direct evolutionary feature extraction algorithm for classifying high dimensional data. *Proceedings, Twenty-first National Conference on Artificial Intelligence (AAAI)*, pages 561–566, 2006.

[11] W. Zheng et al. GA-Fisher: A new LDA-based face recognition algorithm with selection of principal components. *IEEE Transaction on Systems, Man, and Cybernetics - Part B: Cybernetics*, 35(5):1065–1078, 2005.