A two-phase test sample sparse representation method for use with face recognition

Yong Xu a), Member, IEEE, David Zhang b), Fellow, IEEE, Jian Yang c), Jing-Yu Yang c)

a) Bio-Computing Research Center, Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China
b) Biometrics Research Centre, Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong.
c) School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing, China

Abstract In this paper, we propose a two-phase test sample representation method for face recognition. The first phase of the proposed method seeks to represent the test sample as a linear combination of all the training samples and exploits the representation ability of each training sample to determine M 'nearest neighbors' for the test sample. The second phase represents the test sample as a linear combination of the determined M nearest neighbors and uses the representation result to perform classification. We propose this method with the following assumption: the test sample and its some neighbors are probably from the same class. Thus, we use the first phase to detect the training samples that are far from the test sample and assume that these samples have no effects on the ultimate classification decision. This is helpful to accurately classify the test sample. We will also show the probability explanation of the proposed method. A number of face recognition experiments show that our method performs very well.

Keywords: Face recognition; pattern recognition; sparse representation; transform methods; computer vision

1. Introduction

Transform methods, such as linear [1-5] and nonlinear [6-14] methods, have been widely used in face recognition. The principal component analysis (PCA) [1-3] and the linear discriminant analysis (LDA) [4, 5] are two typical examples of linear transform methods. The widely used kernel PCA [6-9] and kernel LDA [10, 11, 12] are two typical examples of nonlinear transform methods. When transforming samples into a new lower-dimensional space, different transform methods have different goals. For example, the PCA transforms samples into a space where the centers of different classes have the maximum distances. Transform methods usually use the entire set of training samples to obtain transform axes and then project each test and training sample onto the transform axes to produce a representation for the sample. They then compute the distance between the representation of the test sample and that of the training sample, and exploit the distance and a classifier to classify the test sample.

Local transform methods that have very different motivations in comparison with conventional transform methods have also been proposed.. The local transform methods exploit local training samples rather than all of the training samples to produce transform axes. For example, Sugiyama integrated the ideas of the LDA and locality preserving projection (LPP) to generate a local transform method for a problem where the samples in a class are multimodal [13]. Vural proposed to use the local dependencies of samples for classification [14]. Liu et al. showed that a local PCA was preferable to a global PCA for feature extraction [15]. Other attempts to use the relationship between samples in a local region can be found in [16-20]. As local and conventional transform methods all first depend on a transform process to produce a representation for the training and test samples, and then use a classifier to classify the test sample, we refer to them as transform-based sample representation methods.

Recently, a method that addresses classification problems from a novel viewpoint has been proposed. This method addresses classification problems by evaluating representation ability on a test sample of each class. For example, the method proposed in [21] imposes a sparse constraint on local discriminant projections, and requires that the test sample to be sparsely represented by the training samples. The methods in [22-28] first use a sparse linear combination of the training samples to represent the test sample. ‘Sparse’ and ‘sparsely’ imply as follows: when a method uses a linear
2. Two-phase test sample representation (TPTSR) method

In this section, we will present the details of the proposed TPTSR method. We assume that there are \( L \) classes and \( n \) training samples, \( x_1, \ldots, x_n \). If a training sample is from the \( j \)th class (\( j = 1, 2, \ldots, L \)), we take \( j \) as the class label of this training sample.

2.1 The first phase of the TPTSR

The first phase of the TPTSR uses all of the training samples to represent each test sample and exploits the representation result to identify the M nearest neighbors of the test sample from the set of the training samples. It first assumes that the following equation is approximately satisfied:

\[
y = a_1 x_1 + \ldots + a_n x_n, \quad (1)
\]

where \( y \) is the test sample and \( a_i \) \((i = 1, 2, \ldots, n)\) are the coefficients. We can rewrite Eq.(1) into the following equation:

\[
y = X A, \quad (2)
\]

where \( A = [a_1 \ldots a_n]^T \), \( X = [x_1 \ldots x_n] \). \( x_1, \ldots, x_n \) and \( y \) are all column vectors. If \( X \) is a nonsingular square matrix, we can solve \( A \) by using \( A = X^{-1} y \); otherwise, we can solve it by using \( A = (X^T X + \mu I)^{-1} X^T y \), where \( \mu \) is a small positive constant and \( I \) is the identity matrix.

Eq.(1) shows that every training sample makes its own contribution to representing the test sample. The contribution that the \( i \)th training sample makes is \( a_i x_i \). The contribution, in representing the test sample, of the \( i \)th training sample \( x_i \) can be also evaluated by the deviation between \( a_i x_i \) and \( y \), i.e. \( e_i = ||y - a_i x_i||^2 \). \( e_i \) can also be somewhat viewed as a measurement of the distance between the test sample and the \( i \)th training sample. We consider that a small \( e_i \) means that the \( i \)th training sample has a great contribution in representing the test sample. We exploit \( e_i \) to identify the M training samples.
samples that have the M greatest contributions, in representing the test sample and denote them by \( x_1, ..., x_M \). We refer to these samples as the M nearest neighbors of the test sample. Let \( C = \{ c_1, c_2, ..., c_r \} \), a set of some numbers, stand for the set of class labels of the M nearest neighbors. If a nearest neighbor is from the \( j \) th class (\( j = 1, 2, ..., L \)), we take \( j \) as the class label of this nearest neighbor. \( C \) must be one subset of set \{1,2,...,L\} i.e. \( C \subseteq \{1,2,...,L\} \). If no neighbor is from the \( p \) th class, then the number \( p \) must be not an element of \( C \). Consequently, the TPTSR will not ultimately classify the test sample into the \( p \) th class.

### 2.2 The second phase of the TPTSR

The second phase of the TPTSR seeks to represent the test sample as a linear combination of the determined M nearest neighbors and uses the representation result to classify the test sample. This phase assumes that the following equation is approximately satisfied:

\[
y = b_1 \tilde{x}_1 + + b_M \tilde{x}_M,
\]

where \( \tilde{x}_i \) (\( i = 1, 2, ..., M \)) are the identified M nearest neighbors and \( b_i \) (\( i = 1, 2, ..., M \)) are the coefficients. We rewrite Eq.(3) into

\[
y = \tilde{X}B,
\]

where \( B = [b_1 ... b_M]' \), \( \tilde{X} = [\tilde{x}_1 ... \tilde{x}_M] \). If \( \tilde{X} \) is a nonsingular square matrix, we can solve \( B \) by using \( B = (\tilde{X}^{-1})^T y \); otherwise, we can solve it by using

\[
B = (\tilde{X}^T \tilde{X} + \gamma I)^{-1} \tilde{X}^T y,
\]

where \( \gamma \) is a positive constant and \( I \) is the identity matrix. After we obtain \( B \), we refer to \( \tilde{X}B \) as the representation result of our method. We can convert the representation result into a two-dimensional image with the same size as the original sample image.

Since the neighbors might be from different classes, we calculate the sum of the contribution to represent the test sample of the neighbors from each class and exploit the sum to classify the test sample. For example, if all the neighbors from the \( r \) th (\( r \in C \)) class are \( \tilde{x}_1, ..., \tilde{x}_r \), then the sum of the contribution to represent the test sample of the \( r \) th class will be

\[
g_r = b_r \tilde{x}_1 + + b_r \tilde{x}_r.
\]

We calculate the deviation of \( g_r \) from \( y \) by using

\[D_r = \| y - g_r \|^2, r \in C.\]

We can also convert \( g_r \) into a two-dimensional matrix with the same size as the original sample image. When we do so, we also refer to this matrix as a two-dimensional image that stands for the contribution of the \( r \) th class. A smaller deviation \( D_r \) means a greater contribution to representing the test sample. Thus, we classify \( y \) into the class that produces the smallest deviation. In summary, the main steps of the TPTSR are as follows:

**Step 1.** Use the first phase to determine the M nearest neighbors for the test sample.

**Step 2.** Exploit the M nearest neighbors of the test sample to construct Eq.(3) and solve this equation.

**Step 3.** Use Eq.(7) to compute deviation \( D_r \) which is generated from the \( r \) th class, \( r \in C \).

**Step 4.** Classify the test sample into the class that has minimum deviation. In other words, if \( D_q = \min D_r \) (\( q, r \in C \)), the test sample will be classified into the \( q \) th class.

Using the second phase, the TPTSR determines the contribution in representing the test sample of different M nearest neighbors and classifies the test sample into the class that makes the largest contribution. The contribution of one nearest neighbor is mainly associated with the corresponding coefficient. If the \( i \) th nearest neighbor has a large similarity with the test sample, then the corresponding coefficient probably has a large absolute value. As a result, the \( i \) th nearest neighbor probably also makes a large contribution to representing the test sample.

### 3. Analysis of our method

This section analyzes the ideas and rationale of the TPTSR and compares it with other methods.

#### 3.1 Methodology comparison between the TPTSR and other methods

In this section, we first present the sparse representation method proposed in [22, 23]. This method obtains promising performances in face recognition. As presented earlier, the method in
[22, 23] proposes to represent the test sample as a sparse linear combination of all the training samples. We note that apart from classification problems, the sparse representation method can be also applied to clustering [27] and motion segmentation problems [28]. In [22-28], the extent of sparseness is evaluated by the $l_1$ norm of the coefficient vector of the linear combination and it is considered that a smaller $l_1$ norm means a higher extent of “sparseness”. The method in [22,23] uses a multi-objective programming algorithm to obtain its solution and has a very high time complexity. In addition, it is almost impossible for the algorithm to simultaneously achieve minimum deviation and $l_1$ norm.

The TPTSR can be also regarded as a supervised sparse representation method. The sparseness is reflected by the following fact: when we rewrite the linear combination of the second phase of the TPTSR as a linear combination of all the training samples, the training sample that is not one of the M nearest neighbors indeed has a coefficient of zero. “Supervised” means that the TPTSR uses the first phase to produce the sparseness, and the sparse extent of the coefficients (i.e. the number of zero coefficients) and which coefficients of the linear combination are zero are known. However, as for the sparse representation method proposed in [22,23], it is not known which coefficients of the linear combination are equal or close to zero. We refer to this method as the unsupervised sparse method.

The TPTSR has both differences and similarities with local transform methods. The TPTSR and local transform methods are similar in that they only use a subset of the training samples. Local transform methods such as the local Fisher discriminant analysis proposed in [13] are able to exploit the local structure of patterns to make a good classification decision. When the TPTSR uses only the M nearest neighbors to represent and classify the test sample, it has the following underlying rationale: the test sample and its nearest neighbors usually belong to the same class. It seems that the k nearest neighbor classifier has a similar idea; however, this classifier exploits the neighbors of the test sample too coarsely (it simply counts the number of class labels of the nearest neighbors and classifies the test sample into the class that contains the most neighbors) and usually produces a bad classification result when k has a large value. When the TPTSR uses a linear combination of the M nearest neighbors to represent and classify the test sample, it specifically takes the relationship between the test sample and neighbors into account by setting different coefficients for different neighbors.

### 3.2 Probability explanation

The classification rule of our method can be described as follows: let $c_1, \ldots, c_L$ denote the $L$ classes, respectively. The first phase of the TPTSR can be explained as a phase that exploits $e_t$ to evaluate the probability that the test sample $y$ and the $ith$ training sample are from the same class. Let $p(T_i \mid y)$ denote this probability. $T_i$ stands for the event that $y$ and the $ith$ training sample are from the same class. The first phase of the TPTSR indeed assumes that $p(T_i \mid y) \propto e^t_{max} - e_i$, where $e^t_{max}$ is used to denote a number greater than the maximum value of $e_i = ||y - a_i x_i||^2$, $i = 1, \ldots, n$. Figure 1 visually shows that $p(T_i \mid y)$ might severely vary with $i$. If a training sample has a small $p(T_i \mid y)$ (shown by the brown bar), the TPTSR will regard that it has no effect on the ultimate classification of the test sample. Actually, the TPTSR does not take this kind of training sample into account and exploits only the remaining training samples to implement the second phase. This implies that the TPTSR reassigns zero to $p(T_i \mid y)$ for this kind of training sample. As the value of $\sum p(T_i \mid y)$ is not variable, the TPTSR indeed implicitly converts Figure 1 into Figure 2, in which the $p(T_i \mid y)$ that originally corresponds to the brown bars in Figure 1 is set to zero, whereas the other $p(T_i \mid y)$ is increased. Figure 2 can be viewed as the probability explanation of the fact that the second phase of the TPTSR neglects some training samples that are far from the testing sample and exploits only the remaining training samples to represent the testing sample.
Figure 1. A possible distribution of $p(T_i \mid y), i = 1, \ldots, n$. The horizontal coordinate shows the number of training samples and the vertical coordinate shows the probability $p(T_i \mid y)()$. A brown bar denotes a small $p(T_i \mid y)$.

Figure 2. A possible modification of $p(T_i \mid y), i = 1, \ldots, n$, shown in Figure 1, made by the second phase of the TPTSR. The $p(T_i \mid y)$ that corresponds to the brown bars in Figure 1 has been set to zero, whereas the other $p(T_i \mid y)$ has been increased.

The second phase of the TPTSR assumes that the probability of test sample $y$ being from class $c_i$, is directly related to $\| y - g_i \|^2$, i.e.

$$p(c_i \mid y) \propto D_{max} - \| y - g_i \|^2, c_i \in C,$$  (8)

Copyright (c) 2011 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.
where $D_{\text{max}}$ stands for a number greater than the maximum value of $\| y - g_i \|^2$. A smaller $\| y - g_i \|^2$ means a greater posterior probability $p(c_i | y)$. The classification rule of the TPTSR indeed classifies the test sample into the class that has the greatest posterior probability. It is known that $C \subseteq \{1, 2, ..., L\}$. Thus, if $c_j$ is an element of $\{1, 2, ..., L\}$ but not an element of $C$, it is impossible for TPTSR to classify the test sample into the $c_j$th class. Actually, in this case, the TPTSR assumes that $p(c_j | y) = 0$. In other words, it is assumed that if one class does not contain any of the $M$ nearest neighbors of the test sample, this class should have a posterior probability of zero, i.e. $p(c_j | y) = 0$.

In contrast to the TPTSR, its global version works in a different way. It uses all the training samples to represent and classify the test sample. As a result, it seems that every $p(c_j | y), j = 1, ..., L$ has a non-zero value. The global version of the TPTSR should identify the maximum posterior probability from all of the $L$ posterior probabilities and classify the test sample into the corresponding class.

As the TPTSR sets some posterior probabilities to zero, it also increases the others. As a result, the TPTSR probably produces a sharper distribution of the posterior probability than its global version. This is shown in Figures 3 and 4. We suppose that in Figures 3 and 4, the test sample is true from the second class. It is possible that due to the interference of the noise, the global version of the TPTSR erroneously classifies the test sample into the sixth class. However, the TPTSR probably correctly classifies the test sample into the second class with a very high posterior probability. Actually, the TPTSR has the following advantage: in a complex case where there is interference caused by noise, the TPTSR is able to achieve a higher accuracy than its global version. As a result, the TPTSR provides us with a good way to deal with complex classification problems, such as face recognition (as there are variations in lighting, facial expression and pose).

4. Experimental results

We conducted a number of experiments on the ORL [30], Feret [31, 32] and AR [33] face databases. The ORL database includes 400 face images taken from 40 subjects, with each subject providing 10 face images.
subjects, the images were taken at different times, with varying lighting, facial expressions (open/closed eyes, smiling/not smiling) and facial details (glasses/no glasses). From the AR face database, we used 3120 gray images from 120 subjects with each subject providing 26 images. These images were taken in two sessions [33]. From the Feret face database, we only used a subset made up of 1400 images from 200 individuals with each subject providing 7 images [31]. This subset is composed of images whose names are marked with two-character strings: ‘ba’, ‘bj’, ‘bk’, ‘be’, ‘bf’, ‘bd’ and ‘bg’.

For each of the ORL and Feret databases, if \( s \) samples of the \( n \) samples per class are used for training and the remaining samples are used for testing, there are \( C_n^s \) possible combinations. As a result, there are \( C_n^s \) training sets and \( C_n^s \) testing sets. For the Feret database, we used 4 images of each subject as the training samples and took the remaining images as test samples. As a result, we tested different methods by using 35 training and testing sample sets from the Feret database. For the ORL database, we tested different methods in two cases. In the first case, we used 5 images of each subject as the training samples and there were 252 training and testing sample sets. In the second case, we used 6 images of each subject as the training samples and there were 210 training and testing sample sets. As the AR face database contained too many samples, we used only the first 8 samples from each class as the training samples and the others as the test samples. Figures 5 and 6 show some of the face images from the AR and ORL databases, respectively. We then resized each face image from the AR database to a 40 by 50 matrix by using the down-sampling algorithm presented in [29]. The face images of the ORL and Feret databases were also resized by using the same algorithm. When implementing our method, we solved \( A \) and \( B \) by using
\[
A = (X^T X + \mu I)^{-1} X^T y \quad \text{and} \quad B = (\tilde{X}^T \tilde{X} + \gamma I)^{-1} \tilde{X}^T y,
\]
respectively. Both \( \mu \) and \( \gamma \) were set to 0.01. We also conducted experiments of PCA, LDA and the sparse method proposed in [23]. All of the PCA and LDA experiments used the nearest neighbor classifier to perform classification.

We refer to \( g_r \) ( \( r \in \{1,2,\ldots,L\} \) ) in Eq.(6) as the reconstruction result of the testing sample, generated from the \( r \) th class. As shown earlier, if \( g_r \) has minimum deviation from the testing sample, then our method classifies the testing sample into the \( s \) th class. We can convert each \( g_r \) into a matrix \( I_r \) with the same size as the matrices of the face images that are directly used in the experiment. \( I_r \) is also referred to as the reconstruction image of the \( r \) th class. Since \( g_r \) is indeed a linear combination of some training samples of the \( r \) th class, it is certain that \( I_r \) also looks like a face image of this class. On the other hand, if the testing sample is from the \( s \) th class and the reconstruction image generated from the class with minimum deviation also looks like a sample from the \( s \) th class, then the method (TPTSR or its global version) will correctly classify the testing sample. Otherwise, the method will erroneously classify the testing sample. As a result, we can easily determine whether a testing sample can be correctly classified by judging whether the reconstruction image generated from the class with minimum deviation from the testing sample looks like the testing sample. We use Figures 7 and 8 to visually show some original face images and reconstruction images.

Figures 7 and 8 show the results of the TPTSR and its global version on a test sample from the Feret database, respectively. It is clear that the TPTSR correctly classifies the test sample, whereas its global version fails to do so. Figure 9 shows the test samples of a subject from the ORL database and the images generated from the representation results of the test sample. The second and third rows of Figure 9 show the representation results obtained using the TPTSR and its global version, respectively. As presented earlier, vector \( \tilde{X}B \) is the representation result obtained using the TPTSR. On the other hand, the representation result obtained using the TPTSR is calculated as follows: first \( A \) is obtained by solving Eq.(2). Then, \( \tilde{X}A \) is taken as the representation result.

Figures 10 and 11 show the means of the classification error rates (%) of our method and its global version. These two figures clearly show that our method is always able to obtain a much lower classification error rate than the global version. For the ORL and AR databases, our method also classifies more accurately than the nearest neighbor classifier. For the Feret database, the maximum rate of the classification errors obtained using our method is close to the mean of the classification error rates obtained using the nearest neighbor classifier. These figures also show that our method can achieve a very low classification error rate when it uses a
suitable number of nearest neighbors to represent the test sample.

Tables 1 to 3 show the classification results of the PCA, LDA and sparse methods. These tables and Figures 10 and 11 clearly show that the TPTSR outperforms the PCA and LDA. In the experiments, we noted that the within-class scatter matrix $S_w$ of the LDA is singular. As a result, we changed it to $S_w + 0.001I$, where $I$ is the identity. These tables and Figures 10 and 11 tell us that the TPTSR can classify much more accurately than the PCA. For example, when the PCA is applied to the Feret database, it obtains a minimum error rate of 36.0%. The TPTSR achieves a minimum error rate of 28.6%, which is 7.4% lower than that of the PCA. Moreover, in all cases, when the TPTSR is applied to the AR database, it obtained a much lower error rate than the LDA. When the LDA is applied to the AR database, the error rate obtained is 34.2%, whereas the TPTSR obtains a minimum error rate of 27.1% and a maximum error rate of 30.1%, which is also the error rate of its global version. In most cases, when the TPTSR is applied to the Feret database, it obtains a lower error rate than the LDA. The minimum error rate of the TPTSR on the ORL database is also lower than that of the LDA. Through the experiments, we also find that more training samples usually result in a lower error rate.

Figure 5. Some face images of a subject from the AR database.

Figure 6. Some face images from the ORL database.

Figure 7. Result of the global version of the TPTSR on a test sample from the Feret database. (a) shows the original test sample. (b), (c), (d), (e) and (f) show the constructed images generated from the five classes that produce the first 5 smallest deviations from the testing sample, respectively. In the figure below, (b), (c), (d), (e) and (f) also stand for the five constructed images with the first 5 smallest deviations, respectively.

Figure 8. Result of the TPTSR on a test sample from the Feret database. (a) shows the same original test sample shown in Figure 7.
Figure 9. The test samples of a subject from the ORL database and images of the representation result of the test sample. The first row shows the original test sample. For the three images in the same column, the first, second and third images are the original test sample, and the images of the representation results obtained using the TPTSR and its global version, respectively. The first five images of each subject are used as the training samples and the others are used as the test samples.

Figure 10. Means of the classification error rates (%) of our method (i.e. the TPTSR), the global version of our method and the nearest neighbor classifier on the ORL face database.

Copyright (c) 2011 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.
Figure 11. Means of the classification error rates (%) of our method, the global version of our method and the nearest neighbor classifier on the AR and Feret face databases.

Table 1  The mean of the classification error rates of the PCA, LDA and sparse methods on the AR database. PCA(50), PCA(100), PCA(150) and PCA(200) indicate that PCA used 50, 100, 150 and 200 transform axes for feature extraction, respectively. LDA (119) means that LDA used 119 transform axes for feature extraction. Tables 2 and 3 shows the method and number of transform axes used in the same way.

<table>
<thead>
<tr>
<th>Method and number of the transform axes used</th>
<th>PCA(50)</th>
<th>PCA(100)</th>
<th>PCA(150)</th>
<th>PCA(200)</th>
<th>LDA (119)</th>
<th>The sparse method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of the rates of classification errors</td>
<td>45.3%</td>
<td>42.7%</td>
<td>42.1%</td>
<td>41.9%</td>
<td>34.2%</td>
<td>46.7%</td>
</tr>
</tbody>
</table>

Table 2  The mean of the classification error rates of the PCA, LDA and sparse methods on the Feret database.

<table>
<thead>
<tr>
<th>Method and number of the transform axes used</th>
<th>PCA(50)</th>
<th>PCA(100)</th>
<th>PCA(150)</th>
<th>PCA(200)</th>
<th>LDA (119)</th>
<th>The sparse method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of the rates of classification errors</td>
<td>38.0%</td>
<td>36.4%</td>
<td>36.2%</td>
<td>36.0%</td>
<td>36.3%</td>
<td>31.0%</td>
</tr>
</tbody>
</table>

Table 3  The mean of the classification error rates of the PCA, LDA and sparse methods on the ORL database. AS the sparse method is too time consuming, we randomly selected 50 sets of training samples and the corresponding sets of testing samples to conduct experiments on the sparse method.

<table>
<thead>
<tr>
<th>Method number of the transform axes used</th>
<th>Mean of the rates of classification errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 training samples per class</td>
<td></td>
</tr>
<tr>
<td>PCA(50)</td>
<td>7.2%</td>
</tr>
<tr>
<td>PCA(100)</td>
<td>7.9%</td>
</tr>
<tr>
<td>PCA(150)</td>
<td>7.8%</td>
</tr>
<tr>
<td>PCA(200)</td>
<td>7.6%</td>
</tr>
<tr>
<td>LDA(39)</td>
<td>4.8%</td>
</tr>
<tr>
<td>The sparse method</td>
<td>5.4%</td>
</tr>
<tr>
<td>6 training samples per class</td>
<td></td>
</tr>
<tr>
<td>PCA(50)</td>
<td>5.2%</td>
</tr>
<tr>
<td>PCA(100)</td>
<td>6.0%</td>
</tr>
<tr>
<td>PCA(150)</td>
<td>6.1%</td>
</tr>
<tr>
<td>PCA(200)</td>
<td>6.0%</td>
</tr>
<tr>
<td>LDA(39)</td>
<td>3.7%</td>
</tr>
<tr>
<td>The sparse method</td>
<td>3.5%</td>
</tr>
</tbody>
</table>

5. Conclusions

The first and second phases of the method proposed in this paper make coarse to fine classification decisions for the test samples, respectively. The first phase first expresses the test sample as a linear combination of all the training samples and then uses the representation result to determine the M nearest neighbors of the test sample. The second phase seeks to represent the test sample as a linear combination of the determined nearest neighbors and then exploits the representation result to perform classification. According to the devised classification rule, the coarse classification produced by the first phase will allow the test sample to be ultimately classified into only one of the class labels of the M nearest neighbors. The fine classification produced by the second phase ultimately determines the most suitable class label for the test sample.

The proposed method is a supervised sparse representation method and uses a very reasonable way to determine the sparseness. Our analysis shows that the proposed method has solid theoretical foundations. A large number of face recognition experiments show a good performance of our method. The two-phase classification framework proposed for the first time in this paper seems to be also useful for improving other face recognition methods. In the future, we will explore the applications of this framework on other methods.

Acknowledgements

This article is partly supported by the Key Laboratory of Network Oriented Intelligent Computation, Program for New Century Excellent Talents in University (No. NCET-08-0156), National Nature Science Committee of China under Grant Nos. 61071179, 60973098, 60803090 and 60902099 60975006, the Fundamental Research Funds for the Central Universities (No. 30911710002).
for the Central Universities (HIT.NSRIF, 2009130) as well as the CERG fund from the HKSAR Government and the central fund from The Hong Kong Polytechnic University.

References


[25] Y. Shi, D. O. Dai, C. C. Liu and H. Yan, Copyright (c) 2011 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.


Copyright (c) 2011 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.