Multiple Marginal Fisher Analysis

Zhenyu Huang, Hongyuan Zhu, Joey Tianyi Zhou, and Xi Peng, Member, IEEE

Abstract—Dimension reduction is a fundamental task of machine learning and computer vision, which is widely used in a variety of industrial applications. Over past decades, a lot of unsupervised and supervised algorithms have been proposed. However, few of them can automatically determine the feature dimension that could be adaptive to different data distributions. To obtain a good performance, it is popular to seek the optimal dimension by exhaustively enumerating some possible values. Clearly, such a scheme is ad hoc and computationally extensive. Therefore, a method which can automatically estimate the feature dimension in an efficient and principled manner is of significant practical and theoretical value. In this paper, we propose a novel supervised subspace learning method called multiple marginal Fisher analysis (MMFA), which can automatically estimate the feature dimension. By maximizing the interclass separability among marginal points while minimizing within-class scatter, MMFA obtains low-dimensional representations with outstanding discriminative properties. Extensive experiments show that MMFA not only outperforms other algorithms on clean data, but also show robustness on corrupted and disguised data.

Index Terms—Automatic dimension reduction, graph embedding, manifold learning, supervised subspace learning.

I. INTRODUCTION

IN THE era of big data, it is challenging and crucial to develop effective and efficient methods to explore the latent value from massive data. However, this is a daunting task due to the increasing dimension of data accompanying very sparse useful information along with a large number of unwanted redundancy and noises [1]. Besides, the high dimension also brings an extra computational overhead, i.e., the so-called curse of dimension.

In the past decades, many dimension reduction algorithms [2]–[6] have been proposed to solve the curse of dimension. In general, existing algorithms can be roughly classified into unsupervised methods and supervised methods. Unsupervised methods aim to find a low-dimensional representation of original data without utilizing any label information. The most popular method is probably the principal components analysis (PCA) [2], which preserves the global structure of data with the maximum variance. More recently, the manifold learning methods are proposed to achieve the nonlinear dimension reduction, and typical works include ISOMAP [7], locally linear embedding [3], neighborhood preserving embedding (NPE) [8], Laplacian eigenmaps [9], locality preserving projections (LPP) [10], and their variants [11]–[16]. The key idea of them is to utilize the local manifold structure embedded in the high-dimensional space. The other well-known methods include sparsity preserving projections [17] and L1-graph [18]. Recently, Peng et al. [19], [20] theoretically discovered the connections between nuclear norm and Frobenius norm. Based on the Frobenius-norm representation, the principal coefficients embedding (PCE) method [21] was proposed and has achieved state-of-the-art performance in unsupervised subspace learning. Supervised methods utilize the label information to obtain more discriminative features. The most representative method is perhaps the linear discriminative analysis (LDA) [22], [23], and its variant [24], which aims to minimize the within-class scatter while maximizing the between-class scatter. In [5], Yan et al. showed that most of the aforementioned methods [25]–[28] can be unified into the graph embedding framework. Based on this framework, they proposed a new algorithm called marginal Fisher analysis (MFA) [5], which maximizes the separability between pairwise marginal data points.

Despite the success of these methods, most of them cannot estimate the dimension of feature space in a data-driven way. In general, they obtain the optimal feature dimension by exhaustively enumerating all possible values based on the classification accuracy. Clearly, such a strategy is computationally extensive and may cause the overfitting problem. Recently, some methods have been proposed to solve this problem, e.g., PCE [21] and MFA [5]. PCE reduces the dimension without the help of label information, which could also automatically estimate the dimension. Though PCE achieves impressive results, it is highly desirable to develop supervised automatic dimension reduction method. In practice, however, only a few efforts have been devoted. Under the framework of graph embedding [5], MFA is proposed, which builds two graphs.
based on the marginal data points with the help of labels. One major advantage of MFA is that the feature dimension could be determined by using the number of between-class marginal pairs. However, MFA does not give mathematical detail on the feature dimension range; hence, it is more like a heuristic method. In fact, MFA and its variants [29] barely explore the connection between feature dimension and the number of between-class marginal pairs in theory.

In this paper, we propose a novel supervised dimension reduction method called Multiple Marginal Fisher Analysis (MMFA), which could enjoy the advantage of automatic dimension estimation. Unlike the well-known LDA which assumes that data points follow the multivariate Gaussian distribution [30], [31], MMFA estimates the feature dimension using the marginal data points and the local consistence (i.e., manifold structure), thus, avoiding the requirement of data distribution assumption and enjoying promising performance in practical applications. A comparison between LDA and MMFA is shown in Fig. 1. Furthermore, different from other Fisher rule based methods such as MFA, our method could solve the class-isolation issue, i.e., when some classes are isolated from the others, the embeddings may overlap in the feature space and ignore the separability between the isolated classes and other classes. More details could refer to Fig. 2. In order to overcome this disadvantage, MMFA constructs the between-class graph by considering the multiple marginal data pairs which are also shown in Fig. 2.

**Notations:** For ease of presentation, we first define the used mathematical notation through this paper. To be exact, the lower-case letters denote scalars, the lower-case bold letters denote the vectors, and the upper-case bold ones denote matrices. Besides, for a given matrix \( A \), \( r(A) \) denotes the rank of \( A \) and \( T_r(A) \) denotes the trace of \( A \).

**Organization:** The rest of this paper is organized as follows. In Section II, we briefly introduce some related works. Section III introduces our proposed MMFA. The experimental results are shown in Section IV. Finally, we give the conclusion and further discussion of this paper in Section V.

**II. RELATED WORKS**

In this section, we briefly introduce some related works for dimension reduction including unsupervised method PCE [21] and supervised methods including LDA [23] and MFA [5].

**A. Principal Coefficients Embedding (PCE)**

Recently, Peng et al. [19] have shown that Frobenius-norm-based representation could enjoy the low-rank structure owned by nuclear-norm-based representation. Based on this theoretical study, Peng et al. [21] proposed a novel unsupervised subspace learning method called principal coefficient embedding, which could achieve both robustness and automatic dimension estimation.

For a given data \( X = \{x_0, x_1, \ldots, x_n\} \), PCE aims to removes the noise \( E \) from \( X \) to recover the clean data with self-representation regularization of \( X_0 \). The objective function is given as follows:

\[
\min_{C, X_0, E} \frac{1}{2} \|C\|_F^2 + \frac{\lambda}{2} \|E\|_F^2
\]

s.t. \( X = X_0 + E, X_0 = X_0C \) (1)

where \( C \) denotes the representation matrix, which is obtained by performing SVD on the original data.

After obtaining the representation \( C \), PCE yields the low-dimensional features by embedding \( C \) into the feature space as an invariance. Although PCE has achieved state-of-the-art performance in image feature extraction, it does not utilize available label information to boost the performance for classification tasks.

**B. Linear Discriminant Analysis (LDA)**

Different from PCE, LDA [23] is a supervised subspace learning method, which aims to learn a space in which within-class data points (i.e., the points belonging to the same class) are as close as possible and between-class data points (i.e., the points belonging to different classes) are as far as possible.

For a given data \( X = \{x_0, x_1, \ldots, x_n\} \) distributed over the classes \( \{c_0, c_1, \ldots, c_n\} \), LDA obtains the low-dimensional features \( \hat{Y} \) with the help of the learned projection matrix \( A \).
via $Y = AX$. The objective function is as follows:

$$\arg\max_A \frac{A^T S_B A}{A^T S_W A}$$ (2)

where $S_B$ and $S_W$ denote the between-class and within-class scatter matrix, respectively, with the following definition:

$$S_B = \sum_{i=1}^{n_c} N_i (\hat{x}_i - \bar{x})(\hat{x}_i - \bar{x})^T$$

$$S_W = \sum_{i=1}^{n_c} \sum_{x_k \in X_i} (x_k - \hat{x}_i)(x_k - \hat{x}_i)^T$$ (3)

where $\hat{x}$ denotes the mean vector of $X$, $X_i$ denotes the data set belonging to the class $c_i$, whose mean vector is $\hat{x}_i$, and $N_i$ is the number of samples in $X_i$.

LDA leans discriminative features by utilizing within-class similarity $S_W$ and between-class separability $S_B$. In theory, the maximal feature dimension of LDA is $n_c - 1$ due to the rank of matrix $S_B$ is less than $n_c - 1$. Thus, it would lead inferior performance for a large-scale dataset since $n_c - 1$ features may be insufficient to keep crucial information of the input space as explained in [23].

### C. Marginal Fisher Analysis

Yan et al. [5] have shown that most dimension methods can be unified into a graph embedding framework. Under this framework, the dimension reduction methods obtain low-dimensional features by preserving the graph geometric structure from input space into a feature space. Along with this framework, a new supervised algorithm called MFA was proposed, of which major novelty lies on constructing a between-class and within-class graph as follows.

1) Within-class: $W_{ij} = W_{ji} = 1$ if $x_i$ is among the $k_1$ nearest neighbors (NNs) of $x_j$ in the same class.

2) Between-class: $W_{ij}' = W_{ji}' = 1$ if $(x_i, x_j)$ is among the $k_2$ shortest pairs among the set $\{ (x_i, x_j) | x_i \in X_c, x_j \in X_c \}$.

where $W$ and $W'$ are the affinity matrices which denote the similarity of within-class and separability of between-class, respectively. MFA minimizes the similarity and simultaneously maximizes the separability in the low-dimensional space like LDA.

### III. MULTIPLE MARGINAL FISHER ANALYSIS

In this section, we propose the MMFA, which enjoys three advantages, namely, data-adaptive feature dimension estimation, discriminative feature thanks to available data annotation, and a provable feature dimension lower bound.

#### A. Multiple Marginal Fisher Analysis

Most of the dimension reduction methods could be regarded as preserving the geometric structure and label information which correspond to an affinity graph and penalty graph, respectively. As discussed in Fig. 1, LDA suffered from the limitation of the Gaussian distribution assumption. Then, MFA [5] was proposed to solve this limitation by characterizing the between-class separability which only depends on the marginal data points. However, MFA suffers from the class-isolation issue as shown in Fig. 2. Hence, we propose a novel dimension reduction method called MMFA, which not only applies to the non-Gaussian cases but also solves the class-isolated issue.

For a given $x_i$, we define the corresponding low-dimensional feature $\hat{y}_i$ with the projection matrix $A$ as follows:

$$\hat{y}_i = A^T x_i.$$ (4)

A certain criterion motivated by LDA [23] is to minimize the within-class similarity and maximize the between-class separability in the low-dimensional space. In MMFA, we characterize the within-class similarity in the embedding space by following [9]:

$$S_W = \sum_i \sum_j \|y_i - y_j\|^2 W_{ij}$$

$$= \sum_i \sum_j (y_i^T y_i - 2y_i^T y_j + y_j^T y_j) W_{ij}$$

$$= \left( \sum_i W_{ij} \right) y_i^T y_i + \sum_j \left( \sum_i W_{ij} \right) y_j^T y_j - 2 \sum_i \sum_j y_i^T y_j W_{ij}$$

$$= 2 \sum_i D_i y_i^T y_i - 2 \sum_j \sum_i y_i^T y_j W_{ij}$$

$$= 2 Tr(Y^T D Y) - 2 Tr(Y^T W Y)$$

$$= 2 Tr(A^T X(D - W')X^T A).$$ (5)

Furthermore, the between-class separability $S_B$ is characterized by the multiple marginal pairs as follows:

$$S_B = \sum_i \sum_j \|y_i - y_j\|^2 W_{ij}'$$

$$= 2 Tr(A^T X(D' - W')X^T A)$$ (6)

where $D$ and $D'$ are defined as

$$D_{ij} = \sum_j W_{ij}, \quad D_{ij}' = \sum_j W_{ij}'$$ (7)

where the $W$ and $W'$ are computed from the within-class and between-class data points as follows.

1) Within-class graph: We put an edge on the data points $x_i$ and $x_j$ if $x_i$ is among the $k_1$ NNs of $x_j$

$$W_{ij} = \begin{cases} \|x_i - x_j\|^2, & \text{if } x_i \text{ and } x_j \text{ are connected in the within-class graph.} \\ 0, & \text{otherwise.} \end{cases}$$ (8)
corresponding to the composition problem [32]:

\[ W \text{ the rank of the between-class matrix} \]

MMFA can automatically estimate the dimension with which is ad hoc and computationally extensive. The proposed ating all possible dimension based on the classification accuracy, d automatically determine the feature dimension d.d.

\[ \text{Fig. 3. Illustration on the graph construction of MMFA, where} \]

HUANG et al. theorem.

class graph is built by the shortest pairs among every two classes.

\[ \text{Within-class graph} \]

d 

\[ \text{Between-class graph} \]

To be specific, the optimal projection matrix \( A \) is obtained by maximizing the between-class separability defined in (6)

\[ \text{D. Computational Complexity Analysis} \]

For a given data set \( X \in \mathbb{R}^{m \times n} \), MMFA constructs the aforementioned graphs in \( O((k_1 + k_2 + \frac{n_c(n_c-1)}{2})n^2) \). Finally, MMFA performs eigendecomposition on (11) in \( O(m^3) \). Thus, the time complexity of MMFA is \( O(n_c^2 n_c^2 + m^3) \) due to \( k_1, k_2 \ll n_c^2 \).

**IV. EXPERIMENTS AND RESULTS**

In this section, we compare the proposed MMFA with seven state-of-the-art dimension reduction methods including LDA [23], MFA [5], LDE [33], PCE [21], PCA [2], NPE [8], and NMF [34]. The baseline results without any dimension reduction are also provided.

**Algorithm 1: Multiple Marginal Fisher Analysis.**

**Input:** A given data set \( X = \{x_i\}_{i=1}^n \in \mathbb{R}^{m \times n} \), the label information \( c \), and the nearest neighbor number \( k_1 \) and \( k_2 \) of within-class graph and between-class graph.

1. Construct the between-class separability and within-class similarity matrixes:
   \[ \text{Within-class graph: Each sample} \ x_i, \text{set} \ W_{ij} = \|x_i - x_j\|^2 \text{ or 1 if} \ x_j \text{ is among the} \ k_1 \text{ NNs} \]
   \[ \text{of} \ x_i \text{ in the same class otherwise 0.} \]
   \[ \text{Between-class graph: For every two classes} \ c_a \text{ and} \ c_b, \text{set} \ W_{ij} = \|x_i - x_j\|^2 \text{ or 1 if} \ x_j \text{ is} \]
   \[ \text{the} \ k_2 \text{ shortest pairs among the set} \{\text{shortest pairs among} \}
   \[ \text{the between-class graph.} \]
   \[ \text{Note that here we define the weights by the distance of data pairs. Another simple alternative approach is to define the} \]
   \[ \text{weights by 0 (connected) and 1 (disconnected).} \]
   \[ \text{By maximizing the between-class separability defined in (6) and minimizing the within-class similarity in (5), we propose} \]
   \[ \text{the following objective function:} \]
   \[ \text{arg max}_A \frac{\text{Tr}(A^T X (D - W') X^T A)}{\text{Tr}(A^T X (D - W) X^T A)} \]

\[ \text{which can be solved with the following generalized eigendecomposition problem [32]:} \]
\[ X(D' - W') X^T A_i = \lambda_i X(D - W) X^T A_i. \]

To be specific, the optimal \( A \) consists of the eigenvectors corresponding to the \( d \) largest eigenvalues, i.e.,

\[ A = \{a_0, a_1, \ldots, a_{d-1}\}. \]

**B. Dimension Estimation**

As we have proved that the optimal projection matrix \( A \) consists of \( d \) eigenvectors in (11). A crucial problem is how to automatically determine the feature dimension \( d \). Most of the existing methods find the dimension \( d \) by exhaustively enumerating all possible dimension based on the classification accuracy, which is ad hoc and computationally extensive. The proposed method MMFA can automatically estimate the dimension with the rank of the between-class matrix \( W' \) using the following theorem.

**Theorem 1:** For a given data set \( X \), the feature dimension \( d \) can be estimated by the rank of \( D' - W' \), i.e.,

\[ n - k_2 \times n_c \leq d \leq \min (m, n). \]

Theorem 1 helps determine the feature dimensions and the feature dimension set to the lower bound in our experiments.
A. Experiment Settings and Datasets

We carry out experiments on three real-world datasets including AR facial database [35], extended Yale dataset B [36], CASIA-3D FaceV1,1 and UPSPS dataset.2 To evaluate the performance of the tested methods, we use the extracted features for classification and accuracy as the performance. The used datasets are as follows.

AR face images: The used AR dataset [21] contains three subsets. One contains 1400 clean faces of 100 subjects with different facial expressions and illuminations. The other two subsets are disguised by sunglasses or scarves, both of them contain 600 samples of 100 subjects. Each image is with the size of $55 \times 40$.

Extend Yale B face image: The used dataset [21] contains 2204 samples of 38 subjects (58 samples each) and all images are cropped to the size of $54 \times 48$.

CASIA-3D FaceV1: The dataset contains 4624 samples of 123 subjects under different illumination, expression, and poses. In the experiment, we use all front faces which contain 1000 images from 100 subjects (10 samples each). All the images are with the size of $60 \times 50$.

USPS digits: The dataset contains 11 000 samples of 10 digits ($0 \sim 9$). All the images are with the size of $16 \times 16$.

Like [21], we employ the NN classifier to investigate the performance of these feature extraction methods in terms of classification accuracy and time cost. Note that, MMFA, LDA, MFA, and PCE can automatically estimate the feature dimension with different values. We set $d = n - k_2 \times n_c$ in MMFA as described in Section III. Following the experiment settings in [21], all non-adaptive methods reduce the dimension to 300. Like in [5] and [23], we first perform PCA on the input data to preserve $n - n_c$ dimension to avoid the singular problem before MMFA, MFA, and LDA. In experiments, we report the best results by exploring some possible parameter values. More specifically, we set $k_1$ and $k_2$ of MMFA between 1 and $n_c$, where $n_c$ denotes the number of samples for each class. Note that we have provided two weight definition choices: 0/1 (connected/disconnected) or the distance of connected pairs. In the following experiments, we use MMFA1 to denote the first method and MMFA2 denotes the latter one.

For all the evaluated methods, we report the mean and standard deviation of classification accuracy over five randomly sampling data partitions.

B. Performance on Clean Data

In this section, we report the experimental results on the clean datasets including AR, extend Yale B, and CASIA. In order to investigate the influence of different ratio between training and testing size, we randomly split each dataset into two parts with different training–testing ratio. The training/testing data size is denoted by $S_1/S_2$, where $S_1$ denotes the samples of each subject in training data and $S_2$ denotes the samples of each subject in testing data.

In the experiments, we employ the NN classifier to evaluate the dimension reduction performance. Both the classification accuracy and time costs are reported in Tables I–III from which one could observe the following.

1) In most cases, MMFA remarkably outperforms the other methods on the three datasets with the NN classifier.

2) For the different training and testing size, MMFA outperforms the baselines on AR and CASIA. On extend Yale B, MMFA obtains better results in the case of 29/29 and 10/48, and is competitive to LDE.

3) Though MMFA considers the multiple marginal pairs, the computation time increases a little as one could see in the tables.

C. Performance on Corrupted and Disguised Images

In this section, we evaluate the robustness of MMFA against corrupted and disguised images, which are as follows.
**TABLE II**

<table>
<thead>
<tr>
<th>$S_1 / S_2$</th>
<th>29/29</th>
<th>15/43</th>
<th>10/48</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithms</strong></td>
<td><strong>Accuracy (%)</strong></td>
<td><strong>Time (s)</strong></td>
<td><strong>Para.</strong></td>
</tr>
<tr>
<td>Baseline</td>
<td>66.64±1.46</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MMFA1</td>
<td>98.38±0.44</td>
<td>21.49±1.30</td>
<td>18, 10</td>
</tr>
<tr>
<td>MMFA2</td>
<td>98.11±0.47</td>
<td>22.81±1.12</td>
<td>22, 9</td>
</tr>
<tr>
<td>MFA</td>
<td>76.27±1.80</td>
<td>14.78±1.13</td>
<td>1, 780</td>
</tr>
<tr>
<td>LDA</td>
<td>97.96±0.35</td>
<td>27.84±2.18</td>
<td>37</td>
</tr>
<tr>
<td>LDE</td>
<td>98.16±0.35</td>
<td>22.68±0.69</td>
<td>8, 100</td>
</tr>
<tr>
<td>PCE</td>
<td>96.33±0.38</td>
<td>15.04±0.14</td>
<td>15</td>
</tr>
<tr>
<td>PCA</td>
<td>77.36±1.34</td>
<td>14.90±1.12</td>
<td>-</td>
</tr>
<tr>
<td>NPE</td>
<td>89.78±1.29</td>
<td>20.07±0.77</td>
<td>288</td>
</tr>
<tr>
<td>NMF</td>
<td>83.68±2.28</td>
<td>88.71±0.63</td>
<td>-</td>
</tr>
</tbody>
</table>

*Note: The significant level is fixed to 0.05.*

**TABLE III**

<table>
<thead>
<tr>
<th>$S_1 / S_2$</th>
<th>5/5</th>
<th>4/6</th>
<th>3/7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithms</strong></td>
<td><strong>Accuracy (%)</strong></td>
<td><strong>Time (s)</strong></td>
<td><strong>Para.</strong></td>
</tr>
<tr>
<td>Baseline</td>
<td>83.63±2.17</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MMFA1</td>
<td>91.40±1.70</td>
<td>20.05±2.05</td>
<td>1, 1</td>
</tr>
<tr>
<td>MMFA2</td>
<td><strong>91.44±2.18</strong></td>
<td>21.23±1.87</td>
<td>1, 1</td>
</tr>
<tr>
<td>MFA</td>
<td>90.24±1.47</td>
<td>17.49±1.14</td>
<td>2, 120</td>
</tr>
<tr>
<td>LDA</td>
<td>90.99±1.27</td>
<td>16.71±0.49</td>
<td>99</td>
</tr>
<tr>
<td>LDE</td>
<td>90.84±1.12</td>
<td>23.15±3.79</td>
<td>1, 50</td>
</tr>
<tr>
<td>PCE</td>
<td>90.42±1.19</td>
<td>24.83±0.12</td>
<td>20</td>
</tr>
<tr>
<td>PCA</td>
<td>91.08±1.68</td>
<td>18.13±0.91</td>
<td>-</td>
</tr>
<tr>
<td>NPE</td>
<td>91.24±1.61</td>
<td>4.07±0.35</td>
<td>95</td>
</tr>
<tr>
<td>NMF</td>
<td>74.40±3.69</td>
<td>80.24±0.43</td>
<td>-</td>
</tr>
</tbody>
</table>

*Note: The significant level is fixed to 0.05.*

Fig. 4. Some samples from AR faces and extended Yale B faces with Gaussian noise, where the noise ratio increases from 10 to 30%.

1) **Corrupted Data:** First, we investigate the performance of MMFA on the AR faces and extend Yale with Gaussian noise which is the most common-seeing noise in real world. The Gaussian noise is added via $x_i' = x_i + \rho n$, where $\rho$ is the noise ratio, and $n$ denotes the noise following the Gaussian distribution. Fig. 4 shows some sample images with the corruption. In this experiment, we only randomly add Gaussian noise into a half of faces, namely, half of the faces are clean and half of them are corrupted. Similar to the experiments on the clean data, we evaluate the performance of different training/testing size using the NN classifier.

Both the mean and the standard deviation of classification accuracy are reported in Tables IV and V, from which we can see that MMFA is more robust than other methods in the most experiments on AR and extend Yale data.

2) **Disguised Data:** In practice, a large area of images may be corrupted as shown in Fig. 5. In this section, we conduct two experiments with such a case by using disguised AR images. The first experiment is carried out on the AR faces disguised by scarves (occlusion rate is about 40%). The used dataset contains 600 clean samples and 600 disguised samples. The second test is conducted on AR faces disguised by sunglasses (occlusion rate is about 20%), where the dataset contains 600 clean samples and 600 disguised samples. In these two experiments, we randomly generated five different data partitions and each partition contains training and testing subsets with equal size. From Tables VI and VII, one can observe that MMFA outperforms all the baselines on these two disguises.
TABLE IV

PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS USING THE AR FACES CORRUPTED BY GAUSSIAN NOISE

<table>
<thead>
<tr>
<th>Gaussian ratio</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>Accuracy (%)</td>
<td>Time (s)</td>
<td>Para.</td>
</tr>
<tr>
<td>Baseline</td>
<td>37.71±1.19</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MMFA1</td>
<td>89.51±1.60</td>
<td>10.92±1.48</td>
<td>3, 6</td>
</tr>
<tr>
<td>MMFA2</td>
<td>89.48±1.86</td>
<td>11.31±1.32</td>
<td>3, 5</td>
</tr>
<tr>
<td>MFA</td>
<td>88.85±1.30</td>
<td>7.68±0.31</td>
<td>3, 260</td>
</tr>
<tr>
<td>LDA</td>
<td>89.28±1.41</td>
<td>9.48±0.57</td>
<td>99</td>
</tr>
<tr>
<td>LDE</td>
<td>88.37±1.55</td>
<td>10.81±0.75</td>
<td>1, 20</td>
</tr>
<tr>
<td>PCE</td>
<td>86.20±1.27</td>
<td>8.61±0.61</td>
<td>20</td>
</tr>
<tr>
<td>PCA</td>
<td>58.62±2.22</td>
<td>9.39±1.23</td>
<td>-</td>
</tr>
<tr>
<td>NPE</td>
<td>77.37±2.25</td>
<td>6.15±0.54</td>
<td>110</td>
</tr>
<tr>
<td>NMF</td>
<td>54.08±3.37</td>
<td>69.98±0.21</td>
<td>-</td>
</tr>
</tbody>
</table>

Note: All methods except PCE, MMFA, MFA, and LDA extract 300 features for classification. The significant level is fixed to 0.05.

TABLE V

PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS USING THE EXTENDED YALE B FACES CORRUPTED BY GAUSSIAN NOISE

<table>
<thead>
<tr>
<th>Gaussian ratio</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>Accuracy (%)</td>
<td>Time (s)</td>
<td>Para.</td>
</tr>
<tr>
<td>Baseline</td>
<td>67.78±1.09</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MMFA1</td>
<td>95.91±0.30</td>
<td>19.87±1.24</td>
<td>6, 7</td>
</tr>
<tr>
<td>MMFA2</td>
<td>95.89±0.59</td>
<td>20.42±1.74</td>
<td>5, 9</td>
</tr>
<tr>
<td>MFA</td>
<td>73.72±2.55</td>
<td>17.28±1.70</td>
<td>1, 780</td>
</tr>
<tr>
<td>LDA</td>
<td>95.29±0.64</td>
<td>27.37±1.48</td>
<td>37</td>
</tr>
<tr>
<td>LDE</td>
<td>96.17±0.32</td>
<td>24.96±1.98</td>
<td>3, 10</td>
</tr>
<tr>
<td>PCE</td>
<td>95.10±0.50</td>
<td>14.48±0.66</td>
<td>10</td>
</tr>
<tr>
<td>PCA</td>
<td>77.16±1.04</td>
<td>14.77±0.99</td>
<td>-</td>
</tr>
<tr>
<td>NPE</td>
<td>92.86±0.56</td>
<td>10.21±0.38</td>
<td>125</td>
</tr>
<tr>
<td>NMF</td>
<td>67.42±1.41</td>
<td>89.25±0.36</td>
<td>-</td>
</tr>
</tbody>
</table>

Note: The significant level is fixed to 0.05.

Fig. 5. Disguised AR by sunglasses and scarves.

D. Scalability Evaluation

In this section, we give the scalability analysis of MMFA on USPS dataset. In this experiment, we randomly split the dataset into two parts for training and testing, where the number of training samples increases from 500 to 9500 with interval 500. We also compared the other two methods LDA and MFA. The accuracy results are shown in Fig. 6(a). As we can see, the classification accuracy almost remains unchanged when provided 2500 training samples. The computational complexity is $O(n_2^2n_1^2 + n_1^3)$ (Section III-D), which is consistent with the experiment results. To be exact, the increment of computation cost is mostly due to the cost for graph construction, while the embedding cost remains unchanged.

E. Evaluation on Different Classifiers

In order to show the effectiveness of MMFA, we investigate the results of MMFA by using different classifiers compared to other methods. Here, we use three classifiers including NN, support vector machine (SVM), and multilayer perceptron (MLP). Similar to the previous experiments, we randomly split the dataset into two parts with the same size for training and testing. Table VIII shows the results on AR data. As we can see, MMFA outperforms other methods on all the three classifiers.

F. Compared to the Deep Neural Networks

In addition, to show the superiority of our methods, we also compared MMFA to VGG19 network [37], which is pretrained by ImageNet. In the following experiment, we first give the classification accuracy on extracted features obtained by VGG19.
Fig. 6. Scalability analysis of MMFA on the whole USPS dataset, where the training samples increase from 500 to 9500. (a) Classification accuracy of MMFA compared to LDA and MFA. (b) Computation cost of MMFA. Note that the graph time denotes the graph construction cost, and the embedding time denotes the eigendecomposition cost.

Furthermore, we also fine-tune the VGG19 model by adding two full-connection layers to obtain classification results in an end-to-end manner. Note that we retrained the VGG19+fine-tune networks on the training data (i.e., AR, Yale B, and CASIA). We randomly split the dataset into two parts with the same size for training and testing. Table IX shows the results from the VGG19 and VGG19+fine-tune. As we can see, both MMFA1 and MMFA2 outperform VGG19 and VGG19+fine-tune.

G. Influence of Parameters

In this section, we investigate the influence of parameters $k_1$ and $k_2$ of MMFA. Besides the parameters of MMFA, we also report the performance with varying $k$ in the $k$-NN classifier. MMFA characterizes the similarity within-class using $k_1$ neighbors from the same class, while characterizing the separability using $k_2$ shortest marginal pairs among every two classes. In the experiment, we conduct the experiment on the extend Yale B dataset, which is randomly divided into two parts with equal size for training and testing. In other words, the training data contains 1102 samples over 38 subjects (29 samples each). The evaluation setting is as follows.

1) Influence of $k$ in $k$-NN: We investigate the influence of $k$ (the $k$-NN classifier) which ranges from 1 to 28 with fixed $k_1 = 5$ and $k_2 = 5$.

2) Influence of $k_1$: As the training data consist of 29 samples for each subject, we fix $k_2 = 5$ and increase $k_1$ from 1 to 28 according the graph construction strategy.

3) Influence of $k_2$: Similar to $k$, we investigate the performance of MMFA by increasing $k_2$ from 1 to 28 and fixing $k_1 = 5$.

Note that, we fix $k_1$ or $k_2$ to 5 in the above experiment for simplicity. Such a value is not optimal for MMFA.

Fig. 7 shows the influence of parameters. Specifically, Fig. 7(a) shows the performance on the $k$-NN classifier with different $k$. Clearly, MMFA first achieves a competitive result and then becomes worse when $k$ increases from 2 to 4. After that,
Table VIII

Performance Comparison of Different Algorithms With Different Classifiers Using AR Faces

<table>
<thead>
<tr>
<th>$S_1/S_2$</th>
<th>Algorithms</th>
<th>Accuracy (%)</th>
<th>Time (s)</th>
<th>Para.</th>
<th>Accuracy (%)</th>
<th>Time (s)</th>
<th>Para.</th>
<th>Accuracy (%)</th>
<th>Time (s)</th>
<th>Para.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NN</td>
<td></td>
<td></td>
<td></td>
<td>MLP</td>
<td></td>
<td></td>
<td>SVM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accuracy (%)</td>
<td>Time (s)</td>
<td>Para.</td>
<td>Accuracy (%)</td>
<td>Time (s)</td>
<td>Para.</td>
<td>Accuracy (%)</td>
<td>Time (s)</td>
<td>Para.</td>
</tr>
<tr>
<td>MMFA1</td>
<td>92.94±1.45</td>
<td>10.08±0.42</td>
<td>3, 6</td>
<td></td>
<td>90.88±0.70</td>
<td>11.22±0.52</td>
<td>2, 4</td>
<td>96.42±0.37</td>
<td>11.13±0.58</td>
<td>1, 2</td>
</tr>
<tr>
<td>MMFA2</td>
<td>93.20±0.85</td>
<td>9.67±0.61</td>
<td>3, 6</td>
<td></td>
<td>88.77±1.68</td>
<td>10.27±0.20</td>
<td>4, 5</td>
<td>95.71±0.64</td>
<td>11.68±0.91</td>
<td>1, 2</td>
</tr>
<tr>
<td>MFA</td>
<td>92.62±1.00</td>
<td>8.12±0.90</td>
<td>3, 200</td>
<td></td>
<td>90.62±1.55</td>
<td>7.32±0.39</td>
<td>6, 320</td>
<td>95.37±0.92</td>
<td>7.92±0.43</td>
<td>3, 100</td>
</tr>
<tr>
<td>LDA</td>
<td>92.74±1.34</td>
<td>8.62±0.47</td>
<td>99</td>
<td></td>
<td>89.17±0.87</td>
<td>8.51±0.66</td>
<td>99</td>
<td>94.65±0.89</td>
<td>8.81±0.39</td>
<td>99</td>
</tr>
<tr>
<td>LDE</td>
<td>91.54±1.45</td>
<td>11.58±0.88</td>
<td>1, 40</td>
<td></td>
<td>90.08±0.75</td>
<td>10.32±0.69</td>
<td>3, 50</td>
<td>95.94±0.71</td>
<td>10.79±0.93</td>
<td>1, 40</td>
</tr>
<tr>
<td>PCE</td>
<td>87.40±1.89</td>
<td>9.59±0.86</td>
<td>20</td>
<td></td>
<td>88.37±1.20</td>
<td>10.09±0.43</td>
<td>95</td>
<td>94.14±1.07</td>
<td>10.22±0.39</td>
<td>40</td>
</tr>
<tr>
<td>PCA</td>
<td>61.37±1.98</td>
<td>8.94±1.18</td>
<td>-</td>
<td></td>
<td>37.02±2.47</td>
<td>7.21±0.78</td>
<td>-</td>
<td>95.42±0.43</td>
<td>6.95±0.86</td>
<td>-</td>
</tr>
<tr>
<td>NPE</td>
<td>81.42±1.02</td>
<td>5.30±0.33</td>
<td>98</td>
<td></td>
<td>90.71±0.84</td>
<td>4.87±0.72</td>
<td>140</td>
<td>93.99±0.52</td>
<td>5.19±0.53</td>
<td>140</td>
</tr>
<tr>
<td>NMF</td>
<td>61.54±4.49</td>
<td>71.92±0.74</td>
<td>-</td>
<td></td>
<td>88.31±3.19</td>
<td>68.87±1.74</td>
<td>-</td>
<td>90.82±2.5</td>
<td>69.31±0.96</td>
<td>-</td>
</tr>
</tbody>
</table>

Note: The time reported here only includes the dimension reduction cost. The significant level is fixed to 0.05.

Fig. 7. (a) Classification accuracy with varying parameter $k$ in k-NN classifier with $k_1 = 5$ and $k_2 = 5$. (b) Classification accuracy with varying parameter $k_1$ from 1 to 28 by fixing $k_2 = 5$. (c) Classification accuracy with varying parameter $k_1$ from 2 to 28 by fixing $k_2 = 5$. In addition, we also show the feature dimension with varying $k_2$. Note that we use k-NN classifier in the experiment (a), while the NN classifier is used in the experiment (b) and (c).

Table IX

Comparison With VGG19 Networks

<table>
<thead>
<tr>
<th>Methods</th>
<th>AR</th>
<th>Yale</th>
<th>CASIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMFA1</td>
<td>92.94±1.45</td>
<td>98.38±0.44</td>
<td>91.40±1.70</td>
</tr>
<tr>
<td>MMFA2</td>
<td>92.83±1.26</td>
<td>98.11±0.47</td>
<td>91.44±2.18</td>
</tr>
<tr>
<td>VGG19</td>
<td>80.14±1.68</td>
<td>87.96±0.94</td>
<td>82.28±0.89</td>
</tr>
<tr>
<td>VGG19+fine-tune</td>
<td>85.77±1.49</td>
<td>66.63±1.97</td>
<td>73.39±2.56</td>
</tr>
</tbody>
</table>

Note: The significant level is fixed to 0.05.

the classification accuracy gradually increases when $k$ increases from 4 to 13. In general, MMFA is robust to varying number $k$ in k-NN classifier, whose classification performance almost keeps unchanged in the case of $k > 8$. Fig. 7(b) and (c) show the influence of $k_1$ and $k_2$, respectively. As one can see that, the accuracy of MMFA remarkably increases with $k_1$, and then gives a slight change when $k_1$ increases to 4. Regarding to $k_2$, the accuracy of MMFA increases slowly with $k_2$, and a decline took place when $k_2 = 28$. We find an interesting observation that the accuracy first increases greatly and remains unchanged at $k_1$, while the accuracy first increases slowly and decreases greatly at last. The former phenomenon should attribute to that $k_1 = 1$ misses a lot of within-class information, and the latter one may be resulted from that $k_2 = 28$ cannot keep sufficient information to separate heterogeneous data.

V. Conclusion

In this paper, we proposed a novel supervised subspace learning method called MMFA. Unlike the most existing methods, MMFA can automatically estimate the feature dimension and obtain the low-dimensional representation. Extensive experimental investigations showed that our method could achieve the state of the arts in feature extraction for classifying clean, noisy, and disguised images.

Acknowledgment

The authors would like to thank the associate editor and anonymous reviewers for their valuable comments and constructive suggestions to improve the quality of this paper.

References


Zhenyu Huang received the bachelor's degree in computer science from Sichuan University, Chengdu, China, in 2018. He is currently working toward the master's degree in computer science with the College of Computer Science, Sichuan University, Chengdu, China.

His research interests include deep learning and clustering.

Hongyuan Zhu received the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2015. He is currently a Research Scientist with the Institute for Infocomm Research, Agency for Science, Technology and Research, Singapore. His research interests include multimedia content analysis and segmentation, especially image segmentation/co-segmentation, object detection, scene recognition, and saliency detection.

Joey Tianyi Zhou received the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2015. He is a Scientist with the Institute of High Performance Computing, Research Agency for Science, Technology and Research, Singapore. Dr. Zhou was the recipient of the NIPS 2017 Best Reviewer Award, the Best Paper Nomination at ECCV 2016, and the Best Poster Honorable Mention at ACML 2012.

Xi Peng (S’09–M’14) received the Ph.D. degree in computer science from Sichuan University, Chengdu, China, in 2013. He is a Research Professor with the College of Computer Science, Sichuan University, Chengdu, China.

Prof. Peng has served as an Associate/Guest Editor for several journals including IEEE Access and IEEE Transactions on Neural Networks and Learning Systems. He has been a Session Chair for AAAI’17 and IJCAI’18, a Senior Program Committee Member for IJCAI’17, and a Chair to organize a tutorial at ECCV’16 and a special session at VCIP’17.