A Survey of Four Popular Techniques used in Face Recognition

Yi Han
Department of Electrical and
Computer Engineering
Rutgers, the State University of New Jersy

***Abstract*—In this paper we explored four popular techniques used in face recognition. Eigenface and Fisherface are well-known subspace methods for dimension reduction, Support Vector Machine(SVM) is a widely used classifier over the past decade. We also looked into a relatively young but very powerful method, Sparse Representation-based Classification(SRC). Experiments and comparisons were made among these four techniques.**

I. Introduction

Face recognition is a widely and deeply studied topic in computer vision, it has a number of applications including surveillance cameras, secure authentication and human/computer interface etc. Over the past two decades enormous effect has been put in the area, a great many algorithms and their variations have been proposed to deal with challenges in face recognition such as illumination, facial expression and occlusion etc.

In this paper, we discussed four face recognition techniques which are of great importance and been used in real-world applications. Eigenface and Fisherface are all subspace methods, they seek a subspace of the feature space, where projection of feature vectors onto the subspace retain high variance while at the same time reduce computation complexity significantly. Eigenface treats image samples as observations of a random vector, first tries to make random variables in the vector uncorrelated with each other by diagonalizing the covariance matrix with its eigenvectors, it's like fitting the data with a multivariate Gaussian distribution, or in other words, an ellipsoid. it then chooses a subspace spanned by part of the eigenvectors which preserves most of the variance, the process can be viewed as finding a cross section of the ellipsoid through origin which has longer axises, the length of the axises are just eigenvalues of the covariance matrix. Eigenface scatters all the samples in the subspace, without taking class information, which is vital for classification, into consideration, so Eigenface is kind of mixing data together, while it's advantage is at computation complexity: by projecting all data into the subspace, feature length is significantly reduced, so is computing time. Fisherface however, managed to incorporate class information into the object. It tries to maximize between class scatter while at the same time minimize within class scatter. In this sense, samples are clustered according to the class information, thus can be easily classified. Eigenface and Fisherface are actually feature extraction methods, the try to obtain an compact representation of the image. To deal with the recognition task, classifiers are needed.

Support Vector Machine(SVM) is probably the dominant classifier in the last two decades. SVM tries to learn a hyperplane between two different classes of data, which maximize the margin between the plane supported by support vectors of each class and the hyperplane. The hyperplane can be viewed as a decision function thus data from different classes can be well separated. In the linearly inseparable case, kernel functions are involved to map data to a higher, linear separable dimension. However, experimental results shows that SVM relies heavily on choices of parameters and size of training set, improper parameters or small training set may results in unacceptable classification accuracy. Thus a cross validation technique is required to search for the best parameters. Sparse Representation-based classification(SRC) is a relatively young but promising method been studied over the past few years. SRC borrowed the idea of sparse representation from compressed sensing in signal processing field, which represents each query image with samples in the training set, or in other words, transforms each image into the space spanned by samples in the training set. The result vector will only have large coefficients in the entries corresponding to training images of the same class, thus it's very sparse. The process can be viewed as solving an underdetermined linear system of equations, thus the solution is not unique, seeking for the sparest solution, we can get the best representation of each query image corresponding to all images in the training set. Then residual comparisons between original images and the image reconstructed by their sparse representations are made to determine the classification results. Experimental result shows that SRC is very power full even with a training set of very small size.

We explored the concepts and theories of these four techniques, tried to look into their advantages as well as drawbacks. We tested these methods on the extended YaleB dataset, looked into the influence of some parameters. The remaining content of the paper is organized as follow: In section 2, we explained the theories of these four methods; In section 3, we gave our experimental results and analysis; In section 4, we draw our conclusions and propose some challenges that might be improved in further studies.

II. METHODS

A. Eigenface

Eigenface depends mainly on Principal Components Analysis(PCA). PCA diagonalizes the covariance matrix of a set of samples which is assumed to be generated from the a same probability distribution by its eigenvectors. These eigenvectors are called the principal components of the data. The diagonal entries of the result matrix is the variance of each random variable in the distribution. it can be shown that when arranged in descending order, only a few variance in the beginning can take contain nearly all the information.

Suppose we have a set of samples $\boldsymbol{x}_i$, $i = 1, ..., n$, the covariance matrix S is computed as follow

$$S = \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T \tag{1}$$

where $\boldsymbol{\mu}$ is the sample mean of all the data. if we put all the data in a matrix:

$$B = \begin{bmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu} & \boldsymbol{x}_2 - \boldsymbol{\mu} & ... & \boldsymbol{x}_n - \boldsymbol{\mu} \end{bmatrix} \tag{2}$$

the equation can be written as follow

$$S = BB^T \tag{3}$$

We seek a projection matrix W, which transforms all samples to a space spanned by the column of it

$$\boldsymbol{y} = W^T \boldsymbol{x} \tag{4}$$

if we assume W to be eigenvectors of S and compute the covariance matrix D of the new random vector y, we can easily find out that its just the diagonal matrix mentioned above:

$$D = (W^T \boldsymbol{x})(W^T \boldsymbol{x})^T = W^T S W \tag{5}$$

thus columns of W are the principle components, or eigenfaces if we reshape them to be a image(Fig. 1). By diagonalizing the covariance matrix S, we make the data uncorrelated. Note that the *Total Variance* of the data, that is sum of variance within each dimension, remains unchanged before and after diagonalization since $tr(S) = \sum \lambda_i$, where λ_i are eigenvalues of S.

As mentioned above, most variance of the data is contained by a few biggest principal components, thus to achieve better computation efficiency, we take only these big principal components to construct the projection matrix. In an optimization perspective, this can be expressed as

$$W_{opt} = \arg\max_{W} |W^T S W| \tag{6}$$

where W_{opt} is a $n \times k$ matrix where n is the dimension of the data and k is the number of principal components we want to take. The solution of this optimization problem is shown to be just taking eigenvectors corresponding to largest k eigenvalues of S. Experiments shows that less then 3% of the principal components contains more than 90% of the variance, thus computing time in further classification tasks is significantly reduced.

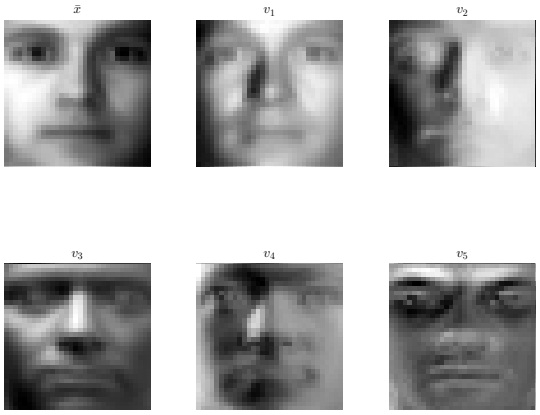

Fig. 1: Visualization of eigenfaces. Largest three eigenvectors correspond to variation due to lighting.

It is almost always the case in face recognition task that the dimension of each sample is very large, covariance matrix S therefore is huge. Doing eigenvalue decomposition on such matrices is time consuming. However, if we look at the definition of the covariance matrix, $S = BB^T$, and the key idea in Singular Value Decomposition(SVD)

$$\sigma_i \boldsymbol{v_i} = B \boldsymbol{u_i} \tag{7}$$

where $\boldsymbol{u_i}$ is eigenvectors of $A^T A$ while $\boldsymbol{v_i}$ is eigenvectors of AA^T, and in our case σ_i is eigenvalues of S, we can easily find out that by doing eigenvalue decomposition on $B^T B$ instead of BB^T, and compute computing eigenvalues of S by $\boldsymbol{v_i} = B\boldsymbol{u_i}/\sigma_i$, we can reduce the size of the covariance matrix from dimension of samples to number of samples in the training set, in most case this is a significant reduction.

The main drawback of Eigenface is none other than the way it projects samples: Eigenface did not take class information into consideration, it tries to maximize the scatter of all samples while to make data more discriminative, we want within class scatter to be small. discarding such this, Eigenface is kind of preserving unwanted information which corresponding to illumination changes between images from the same class [1](illumination changes can be viewed as a translation in the sample space, which separates samples within the same class). Thus data are not well clustered, or even smeared together, making them no longer linearly separable. Therefore, the Main accomplishment of Eigenface is reducing the computation complexity, not improving the discriminative power, we can not expect a higher classification result since the projection process along is losing variances. By discarding the first three principal components, we can expect a better performance since the influence of illumination has been removed [2]. As mentioned above, PCA is trying to fit the data with a ellipsoid, thus Eigenface works well on data with Gaussian distribution,

for data in other distribution format, the result may not be that good.

B. Fisherface

Similar as Eigenface, the Fisherface is also trying to project data onto a subspace. However, unlike Eigenface, which treat every data sample uniformly, Fisherface takes advantage of class information. Thus a better discrimination is achieved between different classes. It's like a supervised way of clustering the data, and consequently classification is simplified.

The Within-class scatter can be defined as

$$S_W = \sum_{ij} (\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)(\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)^T \tag{8}$$

and the total scatter, as defined in last section, is

$$S_T = \sum_{ij} (\boldsymbol{x}_{ij} - \boldsymbol{\mu})(\boldsymbol{x}_{ij} - \boldsymbol{\mu})^T \tag{9}$$

according to these two equation, we can derivate the between class by subtracting within-class from total scatter:

$$\begin{aligned} S_B &= \sum_{ij} [(\boldsymbol{\mu}_i - \boldsymbol{\mu}) + (\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)][(\boldsymbol{\mu}_i - \boldsymbol{\mu}) + (\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i]^T \\ &= \sum_{ij} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T + \sum_{ij} (\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)(\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)^T \\ &\quad + \sum_{ij} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)^T + \sum_{ij} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)^T \end{aligned} \tag{10}$$

for the last two terms in the above equation,

$$\begin{aligned} &\sum_{ij} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)^T \\ &= \sum_{j} (\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i) \sum_{i} (\boldsymbol{\mu}_i - \boldsymbol{\mu}) \\ &= (\sum_{j} \boldsymbol{x}_{ij} - \sum_{j} \boldsymbol{\mu}_i) \sum_{i} (\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \\ &= 0 \times \sum_{i} (\boldsymbol{\mu}_i - \boldsymbol{\mu}) \\ &= 0 \end{aligned} \tag{11}$$

the other term can be computed in the same way. Also,

$$\begin{aligned} &\sum_{ij} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \\ &= \sum_{i=1}^{c} \sum_{j=1}^{N_i} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \\ &= \sum_{i=1}^{c} N_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \end{aligned} \tag{12}$$

thus

$$\begin{aligned} S_T &= \sum_{ij} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T + \sum_{ij} (\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)(\boldsymbol{x}_{ij} - \boldsymbol{\mu}_i)^T \\ &= \sum_{i=1}^{c} N_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T + S_W \end{aligned} \tag{13}$$

the total scatter can be split into within-class scatter and between-class scatter, thus the between-class scatter is defined as

$$S_B = \sum_{i=1}^{c} N_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \tag{14}$$

However, the above way of defining between-class scatter in unbalanced, classes with larger number of samples will dominant the result. To deal with this, we might consider a balanced formula

$$S_B^* = \bar{n} \sum_{i} (\boldsymbol{\mu}_i - \boldsymbol{\mu}^*)(\boldsymbol{\mu}_i - \boldsymbol{\mu}^*)^T \tag{15}$$

where $\bar{n} = \sum_i N_i / c$ is averaging the class size, $\boldsymbol{\mu}^* = \sum_i \boldsymbol{\mu}_i / c$ is averaging the class centres. S_T^* and S_W^* can be defined in a similar way. Note that if each class contains same number of training samples, these two formula coincide.

One way from a optimization perspective of expressing the concept of maximizing the between-class scatter while at the same time minimizing the within-class scatter is

$$W_opt = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|} \tag{16}$$

it is worth noticing that the object function is invariant to scaling of $W : W \to \alpha W$, thus we can alway set the denominator to 1, convert the optimization problem to

$$\begin{aligned} \arg\max_W \quad & -\frac{1}{2} W T S_B W \\ \text{s.t.} \quad & W^T S_B W = 1 \end{aligned} \tag{17}$$

the lagrangian of the problem is

$$L_p = -\frac{1}{2} W^T S_B W + \frac{1}{2} \lambda (W^T S_W W - 1) \tag{18}$$

note that both $\frac{1}{2}$ are added for convenience. Set the first orer derivative to zero, we get

$$S_B W = \lambda S_W W \tag{19}$$

Left multiply W^T on both side of the equation, and solve for λ, we get

$$\lambda = \frac{|W^T S_B W|}{|W^T S_W W|} \tag{20}$$

It's just the object function of the original optimization problem. Eq. 19 is a generalized eigenvalue problem, thus solving the optimization problem is equal to find the eigenvectors corresponding to the largest k eigenvalues of the generalized eigenvalue problem if we restrict the number of column of W to be k.

The maximum rank of S_B is $c - 1$

$$\begin{aligned} &rank(\sum_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T) \\ &= rank(\sum_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^T) - rank(\sum_i \boldsymbol{\mu}_i \boldsymbol{\mu}^T) - \\ &\quad rank(\sum_i \boldsymbol{\mu} \boldsymbol{\mu}_i^T) + rank(\sum_i \boldsymbol{\mu} \boldsymbol{\mu}^T) \\ &= c - 1 - 1 + 1 \\ &= c - 1 \end{aligned} \tag{21}$$

Thus the upper bound of number of eigenvalues of the generalized eigenvalue problem is $c-1$, which means information is well concentrated in a few dimensions.

Most cases in face recognition, due to insufficiency of training samples, we are confronted with the difficulty that the within-class scatter S_W is singular(rank at most be $N-c$) due to insufficiency of training samples compared to number of image pixels. Therefore the generalized eigenvalue problem cannot be solve by simply computing the inverse of S_W, then doing eigenvalue decomposition on $S_W^{-1}S_B$. To deal with this problem, we usually do a PCA first to reduce to dimension of the samples, then the optimal projection matrix can be written as

$$W_{opt} = W_{pca}W_{lda} \tag{22}$$

C. Support Vector Machine

Support Vector Machine is perhaps the most popular classification tool in the past twenty years, before Neural Network stages its comeback.

1) Linear classification: In the simplest linearly seprable case, SVM tries to learn a hyperplane

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b \tag{23}$$

that serves as a decision plane, which separate samples in two different classes. SVM belong to the class of maximum margin classifiers [3]. It looks for the hyperplane that maximize the margin between the support plane of each class with itself. The primal problem model can be expressed as

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1, ...l. \end{aligned} \tag{24}$$

where the latter term is added for tolerance of error, or soft-margin to be precise [4]. The dual problem is

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{1^T}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \alpha_i \leq C,\ i = 1, ...l. \end{aligned} \tag{25}$$

where Q is the $l \times l$ positive semidefinite matrix, $Q_{ij} = y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$. The dual problem can be derivated from the primal problem by apply the method of Lagrange multipliers to combine the object function and the constraints and set its 1st order derivative with respect to $\boldsymbol{w}$ and b to 0.

While most of the techniques used to train classifiers are based on the idea of minimizing the training error, which is called *empirical risk*, SVM operate on another induction principle, called *structural risk minimization*,which minimizes an upper bound on the generalization error.

2) Kernal SVM: The idea of the decision plane can be easily extended to higher dimensions when the data is not linearly separable. By mapping the data to a higher dimensional space $\boldsymbol{z} = \phi(\boldsymbol{x}_i)$, a hyperplane may be found to separate the data. Thank to kernel methods, we don't have to compute the mapping for each training samples but compute the kernel function instead

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j) \tag{26}$$

Consequently the hyperplane, or the decision function can be expressed as

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) + b \tag{27}$$

Commonly used kernel functions are polynomial, radial basis function, sigmoid.

3) Multi-class classification: For multi-class classification task, either one-vs-one or one-vs-all approach is applied [3]. In one-vs-one way totally $c(c-1)/2$ SVMs are trained, each separate a pair of classes if we assume the number of classes to be c. In one-vs-all way only c SVMs are trained, however it may suffer from the problem of imbalanced training set.

D. Sparse Representation-based Classification

Sparse Representation-based Classification(SRC) is a relatively young method first proposed by Wright in 2009 [5]. However, the idea of sparse representation is not new at all, in fields such as signal processing, the concept of sparse is also broadly studied.

1) Sparse Representation: The key idea of sparse representation is represent query images with all the samples in the training set, this is equal to solving the following linear system of equations

$$\boldsymbol{y} = A\boldsymbol{x} \tag{28}$$

where $\boldsymbol{y}$ are query images and A is the matrix columns of who is training samples, $\boldsymbol{x}_0$ is the sparse representation we want to obtain. The linear system of equations is usually underdetermined, thus unique solution cannot be achieved. Instead, we seek the sparsest solution using L_0-minimization

$$\begin{aligned} \arg\min_{\boldsymbol{x}} \quad & ||\boldsymbol{x}||_0 \\ \text{s.t.} \quad & A\boldsymbol{x} = \boldsymbol{y} \end{aligned} \tag{29}$$

However, this problem is NP-hard [6]. Theory in compressed sensing shows that if the solution is sufficiently sparse, L_0-minimization can be approximated by L_1-minimization [7]

$$\begin{aligned} \arg\min_{\boldsymbol{x}} \quad & ||\boldsymbol{x}||_1 \\ \text{s.t.} \quad & A\boldsymbol{x} = \boldsymbol{y} \end{aligned} \tag{30}$$

The reason L_2-minimization is not applied is that solutions from L_2-minimization are not sparse at all [5], thus response of the system corresponding to each query image is not well concentrated around the class it belongs to.

Sparse representation is discriminative naturally, as it could select the subset of base vectors which express the input signal most concentrated and automatically reject other less concentrated representation [7].

2) Classification on Sparse Representation: For classification task, we simple reconstruct the image corresponding to each class of train samples, compute the residual of it between the real query image. it's intuitive that reconstruction with the smallest residual is most likely to be the true class the query image belongs to. The process can be expressed as follow

$$\min_i r_i(\boldsymbol{y}) = ||\boldsymbol{y} - A\gamma_i(\hat{\boldsymbol{x}})||_2 \tag{31}$$

where γ_i is a function that selects the coefficients within the ith class.

Compared with other popular classification methods such as Nearest Neighbour and SVM, SRC classifies test image based on all possible support within the training set [5], while Nearest Neighbour uses only one and SVM uses those on the support plane, thus is expected to have preciser description of query images.

One more thing worth noticing is that once the sparsity is harnessed, the choice of different features is no longer critical [7].

III. Experimental Results

In this section, we apply the aforementioned four methods to the extended YaleB dataset, which contains 38 individuals, each individual have around 60 samples. We simply vectorize each image to form a description. We first look into the overall performance of each method. Then we vary the number of principal components in Eigenface and Fisherface, try to evaluate it's influence on the classification result. Finally, we looked into the influence of size of training set, make comparison to different methods and strategies.

A. Overall Performance

Here we present the best classification result of each method we can reach.We picked 50 training sample per class, and do the testing on the rest, each experiment is repeated 5 times for stability. For the first two subspace method, we applied both a Nearest Neighbour(NN) classifier [8] and SVM [4] to achieve the accuracy, for SVM, a cross-validation process is performed to guarantee a good result, for SRC, we did nothing, just let it run, but the result seems to defeat any other methods.

TABLE I: Classification results over different methods.

Eigenface+NN	Fisherface+NN	SVM
81.13%	95.06%	94.94%
Eigenface+SVM	Fisherface+SVM	SRC
79.56%	94.36%	99.03%

Eigenface achieves the lowest accuracy compared to other methods, no matter simple Nearest Neighbour classifier or advanced SVM is applied, this coincides with our deduction that PCA reduces discriminative power between different classes, smears all samples together, thus followed by a SVM classifier, which seeks a hyperplane that linearly separate the data, is worse than simply finding the nearest neighbour. Fisherface shows similar results on both NN and SVM, and the classification accuracy is satisfying. As mentioned above, Fisherface makes use of class label information, thus gains better discrimination, it like shaping the data, making them clustered. SVM along obtains similar results, however, without dimension reduction, the computation complexity is high, especially a cross validation process is applied. Moreover, the cross validation is necessary to obtain a reasonable result, casual choice of some parameters may results in an unacceptable classification accuracy. SRC shows surprisingly high accuracy, it's nearly 100%. SRC seeks all possible support for the representation of query images, owe to the sparsity of the representation, the coefficients is highly concentrated, thus each images is well described and discrimination between different classes is obvious. However, In our experiment, as the size of the training size increased, the computing time of SRC rises significantly even under a parallel computing setup, thus further study might need to be conducted to fit it into a real-world application.

B. Influence of k

In this experiment we test the influence of number of principal components we pick for the project matrix on the classification result. We set the training sample per class to be 50, and again each run of experiment is repeated 5 times for the stability of the results. As shown in Fig. 2, both Eigenface

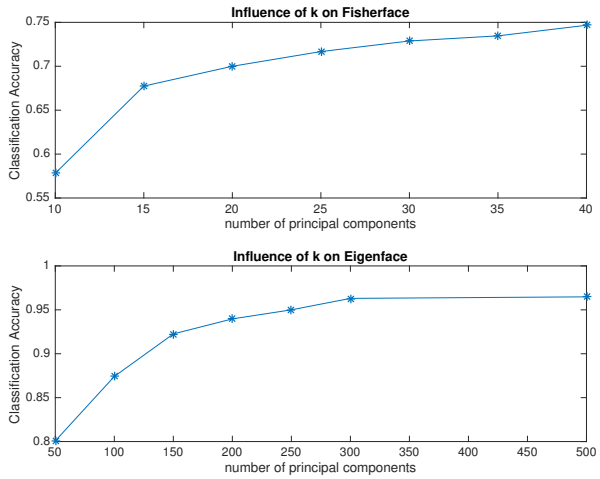

Fig. 2: Influence of number of principal components. A small percentage of principal components is enough to obtain a satisfying classification accuracy

and Fisherface achieve steady results after the number of principal components rises rises. Only a small rise is enough to obtain steady results. Thus by project data onto the subspace of such small dimension, the computation is significantly reduced comparing with the original feature of all image pixels.

C. Influence of training set size

In this experiment we look into the influence of training set size over a variety of methods and strategies. First thing to notice is that when the size of training set is relatively small, the result of Fisherface, no matter with NN or SVM classifier

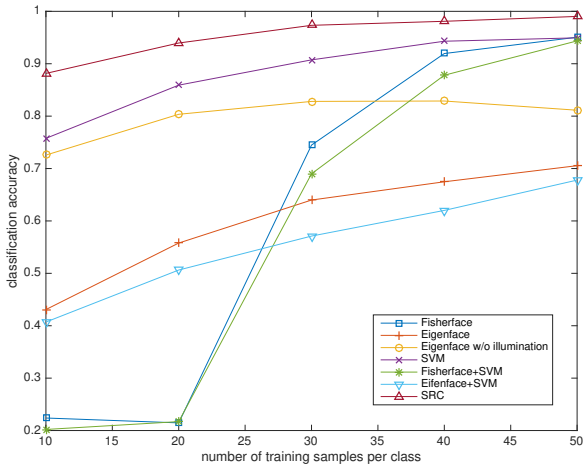

Fig. 3: Influence of training set size.

is much worse even than guess. However, as the number of training sample increases, the classification accuracy increases sharply.

Applying SVM after Eigenface does not do any help to the result, instead, it even worse the situation as discussed above. Removing the largest three principal components in Eigenface results in a increase over all choice of training samples size, which verifies our deduction that by removing the principal components that capture illumination changes, the data is better clustered. The curve of Eigenface with out largest three principal components shows a decrease between 40 and 50, which is also mentioned in [2].

SRC and SVM shows top-level results over all choice of size, they are really reliable classifiers, however again the computational complexity involved in training the model is an issue that need to look into.

IV. Conclusion

In this paper we explored four popular face recognition techniques, briefly introduced their theories, discussed their highlights as well as drawbacks. We did experiment to test the performance of them and some combinations of them. The experiment results shows that SRC outperformed other techniques without any fancy adjustments on parameters or strategies. Thus we can conclude that SRC is really a powerful and promising technique for the face recognition task, however further study might be needed to deal with the issue of computation.

References

[1] Yael Adini, Yael Moses, and Shimon Ullman. Face recognition: The problem of compensating for changes in illumination direction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):721–732, 1997.

[2] Peter N Belhumeur, João P Hespanha, and David J Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, 1997.

[3] Bernd Heisele, Purdy Ho, and Tomaso Poggio. Face recognition with support vector machines: Global versus component-based approach. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 688–694. IEEE, 2001.

[4] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[5] John Wright, Allen Y Yang, Arvind Ganesh, Shankar S Sastry, and Yi Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210–227, 2009.

[6] Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237–260, 1998.

[7] Panqu Wang and Can Xu. Robust face recognition via sparse representation.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.