ECE539: Midterm Project
Face Recognition

Rong Zhang
ECE Department
rz214@scarletmail.rutgers.edu

Abstract

In this project, I implemented four basic face recognition algorithms from the class: Eigenfaces (using PCA), Fisherfaces (using LDA), Support Vector Machine (SVM), and Sparse Representation-based Classification (SRC). To test these algorithms, I used the Extended YaleB dataset to test my algorithms in different numbers of images in training sets and did an analysis on the algorithms of different conditions.

1 Introduction

As a hot topic in deep learning area, face recognition attracts many attentions around the world. There are many approaches to do the image recognition, we can use the subspace methods (PCA, LDA), also can use the classifiers (like SVM). In this project, I implemented four algorithms. These algorithms are: Eigenfaces; Fisherfaces; Support Vector Machine (SVM) and Sparse Representation-based Classification (SRC). In the experiments, I applied the extended YaleB dataset to these algorithms, and calculated the accuracy of these algorithms when the training set are in different size. For different algorithms, there are also some other analysis based on the implementation and the properties of the algorithm.
In this report, I will :

- Introduce these algorithms and dataset.
- Introduce the implementation of the algorithms.
- Discuss and analyze the result of the experiments by the algorithms.

2 Algorithms and Dataset Introduction

2.1 Eigenfaces

Eigenfaces is a face recognition algorithm using PCA, is used to linearly feature dimensionality reduction. Assume X is a n-dimensional feature vectors, after PCA, there will be Y with m-dimensional feature vectors, where $m << n$, and all new feature vectors independent. Eigenfaces use a list of faces vector as training set, and the outputs are some eigenvectors. First, we calculate the mean face using the input dataset, then we calculate ϕ of each image by subtracting the mean image.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\phi_j = x_j - \mu$$

Then, we can find k eigenvectors that best account the distribution of face images within the entire space.

$$w_k : max(\frac{1}{N}\sum_{i=1}^{N}(w_k^T\phi_i)^2), k = 1..m$$

Assuming that $N << n$, we will only need N-1 vectors to represent the dataset. Using the opposite multiplication:

$$B^TBv_i = \mu_i v_i, B = [\phi_1, \phi_2.....\phi_N]$$

$$CBv_i = BB^TBv_i = \mu_i Bv_i$$

So that Bv_i are the eigenvectors of the covariance matrix, where B^TB is an N*N matrix, which is much more manageable. We can calculate the new eigenvectors and keep k of them corresponding to k largest eigenvalues:

$$W = [w_1, w_2....w_k], w_i = Bv_i$$

By compare the projections of test data, we can predict the label of the image.

2.2 Fisherfaces

Fisherfaces is a face recognition algorithm based on LDA. Fisherfaces attempt to maximize the between class scatter, while minimizing the within class scatter. Assuming N is the number of images per class in the databases and c is the number of classes. Average of each class is given by:

$$\mu = \frac{1}{N}\sum_{k=1}^{N} x_k$$

and the class scatter matrix of class i is

$$S_i = \sum_{x_k \in y_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

With-class scatter matrix:

$$S_W = \sum_{i=1}^{c} S_i$$

Between class scatter:

$$S_B = \sum_{i=1}^{c} |y_i|(\mu_i - \mu)(\mu_i - \mu)^T$$

Total scatter:

$$S_T = S_W + S_B$$

We are seeking:

$$W_{opt} = argmax\frac{|W^T S_B W|}{|W^T S_W W|}$$

If S_W is nonsingular, the columns of w:$[w_1, w_2, ..., w_m]$ are the eigenvectors of $S_W^{-1} S_B S_B$. If S_W is singular, apply PCA first to reduce dimensionality of faces:

$$W_{pca} = argmax|W^T S_T W|;$$

$$S_T = \sum_{k=1}^{N}(x_k - \mu)(x_k - \mu)^T$$

Apply FLD on the output

$$W_{fld} = argmax\frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

$$W_{opt}^T = W_{fld}^T W_{pca}^T$$

Now we can compare the projections of the test data to predict the label of the test data.

2.3 Support Vector Machine (SVM)

Support vector machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training images, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a gap as wide as possible. The gap is distance between parallel hyperplanes:

$$\vec{w} \cdot \vec{x} + b = -1$$

and

$$\vec{w} \cdot \vec{x} + b = +1$$

We have

$$D = \frac{2}{\|\vec{w}\|}$$

Because we want to maximize the gap distance, we need to minimize $\|\vec{w}\|$. What's more, we need to impose constrains that all instance are correctly classified.

2.4 Sparse Representation-based Classification (SRC)

Sparse representation represent the test sample in an overcomplete dictionary whose base elements are the training images themselves. If the training images are sufficient for each class, it will be possible to represent the test image as a linear combination of those training images from the same class.

The algorithm is given by:

1. Input a matrix of training samples: $A_i = [A_1, A_2, ..., A_k] \in R^{m*n}$ for k classes, a test sample $y \in R^m$;
2. Normalize the columns of A to have unit l^2-norm.
3. Solve the l^1-minimization problem:

$$\hat{x_1} = argmin_x \|x\|_1, subject\ to : Ax = y.$$

4. Compute the residual $r_i(y) = \|y - A\delta_i(\hat{x}_1)\|_2$, for $i = 1, ..., k$.
5. Output: $identify(y) = argmin_i r_i(y)$.

2.5 The Extended Yale Face Database B

The extended Yale Face Database B used in this project had been modified by the professor. This dataset contains 38 individuals and around 64 near frontal images under different illuminations per individual. At all, there are 2414 images. All the images had been cropped and resized to $32 * 32$. In the experiments, the individual $32 * 32$ image had been flatten to a $1024 * 1$ vector, then use in the algorithms.

3 Implementation, Experiments and Result Analysis

Eigenfaces, Fisherfaces and SVM are implemented in Python. SRC is implemented in Matlab code.

Pre-processing For each algorithm, I randomly chose $p(p = 10, 20, 30, 40, 50)$ images per individual to construct the training set, then use the left images as test set to calculate the prediction accuracy.

3.1 Eigenfaces Analysis

For Eigenfaces, there are several parameters to affect the accuracy of prediction. When we calculated the eigenvalues and eigenvectors of the training set, if we delete one or some eigenvectors corresponding to highest eigenvalues, the accuracy prediction will change. To discuss the effect of this parameter, I did an experiment on this algorithm and deleted $0, 1, 2, 3, 4$ eigenvectors corresponding to highest eigenvalues when $p = 50$ and $k = 200$. Table 1 states the result of this experiment.

# of deleted eigenvectors	Accuracy(%)
0	68.677
1	72.957
2	79.572
3	85.241
4	84.238

Table 1: Prediction accuracy of different numbers of deleted eigenvectors when $p = 50$ and $k = 200$ of Eigenfaces

From this table, I find that deleting 3 eigenvectors will be the best choice to have best prediction accuracy. When we delete 4 eigenvalues, it will lose some accuracy because of losing information of deleting eigenvectors. But deleting 1 to 3 eigenvectors will increase the accuracy of prediction.
With different k values that k is the number of kept eigenvectors, the result accuracy will be different. Usually we don't need all the eigenvectors, because too many eigenvectors will increase the complexity of time and space and there is no obvious increase of the accuracy of prediction. But much too less eigenvectors will also reduce the accuracy of prediction. To find a suitable k values, I did an experiment on different k values. Also, different p value also makes different in the algorithm. With p higher, the size of the training data is larger. To discuss the effect of k and p, I tested this algorithm on several different k values and p values. Table 2 presents the results of this experiment.

	Accuracy(%)				
	$p = 10$	$p = 20$	$p = 30$	$p = 40$	$p = 50$
$k = 20$	58.013	63.845	66.405	67.673	68.397
$k = 50$	67.355	74.607	77.08	78.188	77.432
$k = 100$	71.435	79.262	81.319	81.544	84.825
$k = 200$	73.107	81.499	83.673	83.557	85.241
$k = 300$	74.631	79.504	83.752	85.011	83.269

Table 2: Prediction accuracy of different k and p when deleting 3 eigenvectors of Eigenfaces

When p increased from 10 to 30, there is a relative large increase in the prediction accuracy. However when p is larger than 30, the prediction accuracy is getting stable. For different k values, I find that $k = 100$ or 200 have a good prediction accuracy. When k is less than 100, the number eigenvectors are too small to extract the features of the data. However, $k = 300$ is too large to increase the running time of the algorithm as well as no obvious increase of the prediction accuracy.
Above all the experiments on different parameters, when $k = 200$, $p = 50$, and deleting 3 eigenvectors of largest eigenvalues, Eigenfaces will reach the highest prediction accuracy which is 85.241%.

3.2 Fisherfaces Analysis

For Fisherfaces algorithm, I did the experiments to test the prediction accuracy on different p values. When p is larger, the size of training set is larger and the size of test set is smaller. Table 3 presents the result of this experiment.
From the result table, we can find that Fisherfaces algorithm will reach the highest accuracy of prediction when $p = 50$, and the accuracy is 94.747%.

p	Accuracy(%)
10	80.58
20	85.489
30	82.496
40	93.624
50	94.747

Table 3: Prediction accuracy of different p value of Fisherfaces

As an algorithm using LDA, Fisherfaces show the better performance in extracting the features of input data than Eigenfaces which use PCA.

3.3 SVM Analysis

For SVM implementation, I used scikit-learn, which is a machine learning library for Python. Using scikit-learn can help me to test the effect by different parameters on the prediction accuracy of SVM.
In SVM, there are several kernel functions can be used. I tried all the kernel functions and record the prediction accuracy of these different functions to find the suited distribution of the training data. Table 4 shows the results of this experiments. This experiment is tested when $p = 30$ and no pre-PCA process.

Kernel Choice	Accuracy(%)
Linear	86.18
Poly	66.56
rbf	46.468
Sigmoid	2.904

Table 4: Prediction accuracy of different kernel functions of SVM when $p = 30$

From the result from this experiment, we can conclude that the input data is linearly separable (using the linear kernel function has the best prediction accuracy).
Because the training data size is large, I tried preprocess the training data using PCA to reduce the dimensionality of data to 200 and record the performance. Also, with different p values, I recorded the prediction accuracy of them. Table 5 shows the results of experiments of different p values with or without PCA preprocess.
From this result table, we can find that PCA has no bad influence on the accuracy of prediction. In fact, PCA even denoise the training data for the later SVM classification. So using PCA to preprocess the data will be a good choice to reduce the running time of SVM and keep the performance at the same time. The best accuracy of SVM is 90.856% when p=50 with PCA. What's more, I also tried different values of penalty parameter but it did not have a apparent effect on the predicting accuracy, so I choose the default value 1 for the experiments.

	Accuracy(%)	
p	PCA	No PCA
10	73.894	71.386
20	85.489	84.462
30	88.461	87.677
40	90.604	90.381
50	90.856	89.689

Table 5: Prediction accuracy of different p values with or without PCA pre-process of SVM.

3.4 SRC Analysis

The difficulty in SRC algorithm is to solve the $l1$-minimization problem. For this algorithm, I implemented the Matlab code and use SparseLab toolbox to solve the $l1$-minimization problem. I did experiments on this algorithm to record the prediction accuracy of different p values. Table 6 shows the results of this experiment.

p	Accuracy(%)
10	91.34
20	93.98
30	93.25
40	94.63
50	95.11

Table 6: Prediction accuracy of different p value of SRC

When $p = 50$, the prediction accuracy of SRC is 95.11%, which is the best accuracy I found in this project. This result prove the good performance of SRC over other algorithms. However, $l1$-minimization problem is difficult to solve and the calculation always spend much time, the running time of SRC is very slow (the running time always lasts several hours with individual p value).

3.5 Overall Analysis

In the previous analysis, I presented the results and analysis of each algorithms. In this section, I will discuss these algorithms together. Table 7 shows

the results of these algorithms in different p values. Eigenfaces' results are chosen where $k = 50$ and 3 deleted eigenvectors. SVM's results are chosen with PCA preprocessing.

	Accuracy(%)			
	Eigenfaces	Fisherfaces	SVM	SRC
$p = 10$	73.107	80.58	73.894	91.34
$p = 20$	81.499	85.489	85.489	93.98
$p = 30$	83.673	82.496	88.461	93.25
$p = 40$	83.557	93.624	90.604	94.63
$p = 50$	85.241	94.747	90.856	95.11

Table 7: Prediction accuracy of different p values of different algorithms

Also, I plotted the graph of these algorithms and the graph shows below.

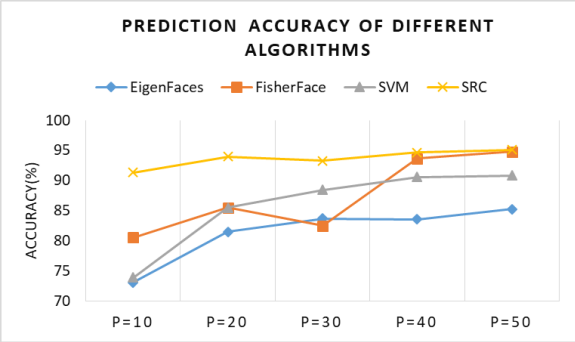

Figure 1: Prediction accuracy of different algorithms

According to the results, SRC has the best prediction accuracy among these algorithms. The accuracy of Fisherfaces is not stable, but usually have a better accuracy compared with Eigenfaces and SVM. Eigenfaces algorithm shows the worst prediction accuracy. The best prediction accuracy is 95.11 from SRC, at the same time Eigenfaces have nearly 10% worse.

The running time for Eigenfaces, Fisherfaces and SVM are much less than SRC. If we want to use SRC in practice, I think a fast solution to l^1-minimization problem is needed. An efficient solver can reduce the running time of SRC effectively. For SVM, applying PCA to the training data will reduce the dimensionality of data, then reduce the time consuming on the classification. For this extended YaleB face dataset, Fisherfaces has a relatively good performance and running time. If you want the best answer, just use SRC and wait for a long time to get the result.

4 Conclusion

In this project, I implemented Eigenfaces, Fisherfaces, SVM and SRC, these four face recognition algorithms and use the extended YaleB dataset to test the algorithm. At the start of this paper, I gave a brief introduction of them. Then for the experiments of each algorithm, I applied different sizes of training data to record the accuracy and also discuss the specific parameters for each algorithms. Among these algorithms, Eigenfaces show the worst prediction accuracy which is 85.241% when $p = 50$ and SRC has the best one which is 95.11% when $p = 50$.

Reference

1. Lecture Slides for Advanced Topics in DSP, ECE539
2. scikit-learn, http://scikit-learn.org
3. SparseLab, https://sparselab.stanford.edu
4. Robust Face Recognition via Sparse
5. Eigenfaces for Recognition, M. Turk and A. Pentland, 1991 Representation: John Wright, Allen Y. Yang, Member, Arvind Ganesh, S. Shankar Sastry, and Yi Ma, 2009
6. Wikipedia: Support Vector Machine, https://en.wikipedia.org/wiki/Support_vector_machine