Midterm Project: Face Recognition

Feng Rong (fr214)

Abstract

Nowadays, face recognition is a hot topic in deep learning, which is widely used in criminals searching, advertisements adapting and album organization. Recognition includes two major parts. One is preprocessing, as known as training process. In training process, it has to train the training images and extract features to build a model. Then it comes to the recognition process, which use the existing model to test the test images and get the outputs. In this project, I build four different model to accomplish the face recognition task, which are eigenfaces(PCA), fisherfaces(LDA), support vector machines(SVMs) and sparse representation-based classification(SRC).

1 Methods Introduction

For this section, I introduced the basic idea of the four methods, that I used to build the face recognition problem. And also gave the formulations of each methods.

1.1 Eigenfaces(PCA)

Eigenfaces, also known as 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 have a list of faces as training set, and the outputs are eigenvectors. First, we calculate the mean face from the dataset, then we align each image from the dataset, by subtracting the mean image.

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

$$\phi_j = x_j - \mu$$

Then, using PCA, we can find orthonormal vectors 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:

$$W^T W v_i = \mu_i v_i$$

$$CW v_i = W W^T W v_i = \mu_i W v_i$$

where $W^T = [\phi_1, ..., \phi_N]$

We can see that Wv_i are the eigenvectors of the Covariance matrix, where $W^T W$ is an N*N matrix, which is much more manageable. Therefore, the new calculation of the eigenvectors is as follows:

$$u_l = \sum_{i=1}^{M} v_{li}\phi_i, l = 1...M(M < N)$$

Although PCA projection is optimal for reconstruction from a low-dimensional basis, it may not be optimal for discrimination, since it assumes that the data has a Gaussian distribution.

1.2 Fisherfaces(LDA)

Fisherfaces attempt to maximize the between class scatter, while minimizing the within class scatter. Assuming that N is the number of images per person in the databases and c is the number of persons in the database. Average of each class:

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

and the between-class scatter matrix 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

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

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

1.3 Support Vector Machines

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, 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 clear gap that is 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 know that

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

Since we want to maximize the gap, we need to minimize $\|\vec{w}\|$.In addition, we need to impose constrains that all instance are correctly classified.

$$y_i(\vec{w} \cdot \vec{x} + b) \geq 1$$

1.4 Sparse Representation-based Classification

Using sparse representation to perform classification, it represent the test sample in an overcomplete dictionary whose base elements are the training samples themselves. If sufficient training samples are available from each class, it will be possible to represent the test sample as a linear combination of just those training samples from the same class.

The complete recognition procedure is shown as follow:
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:

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

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 Implement

For this section, I will introduce the given dataset and my own implementations on the four methods I discussed before.

2.1 Datasets

For the datasets, I used the given datasets Extended YaleB dataset. The original images in the Extended YaleB dataset have been cropped and resized to 32 32. This dataset has 38 individuals and around 64 near frontal images under different illuminations per individual. I was asked to randomly

select (p = 10, 20, 30, 40, 50) images per individual with labels to form the training set, and use the rest of the dataset as the testing set. Apply my algorithms on each of these five splits and record the corresponding accuracies.

2.2 Eigenfaces

For the eigenfaces algorithm, I implemented and evaluated it using python. And for the evaluation part, I did two part. One is with all principal components and another is deleting the first three largest. The source code is attached in the project file.

2.3 Fasherfaces

For the fasherfaces algorithm, I implemented and evaluated it using python. The source code is attached in the project file.

2.4 SVM

For SVMs algorithm, I implemented and evaluated it using python and import a library named "scikit-learn" to build the SVM model. I tried different Kernels to build the SVM model. The source code is attached in the project file.

2.5 SRC

For SRC algorithm, I implemented and evaluated it using MATLAB and import a library named CVX. And the basic SRC model is using a exist example. The source code is attached in the project file.

3 Results Analysis

For this section, I show the results of accuracy of each algorithm with different p. And plotted the accuracy vs number of trainings samples curves overall comparison among all four algorithms. And I also analyzed the results.

3.1 Eigenfaces

For eigenfaces, I have two model. One is with all the principal components and another is deleting three highest. Results is shown as follow:

p	All Eigenvector	Delete 3 highedst
10	43.85%	71.92%
20	55.08%	80.77%
30	61.46%	83.44%
40	66.89%	81.54%
50	69.65%	83.07%

Figure 1: Eigenfaces

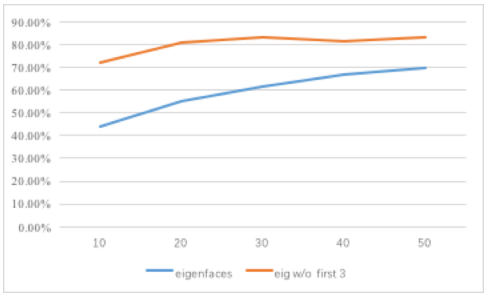

Figure 2: Eigenfaces

We can see from the results that removing the initial three principal components improves the performance of the eigenface method in the presence of lighting variation. And the best accuracy is 83.44% .

3.2 Fisherfaces

For fisherfaces, the results is shown as follow:

p	10	20	30	40	50
Accuracy	79.01%	85.55%	80.85%	91.94%	93.19%

Figure 3: Fisherfaces

As we can see from the results, the accuracy tends to be higher while p grows. And the overall performance of fasherface is better than eigenface.

3.3 SVM

For the SVM, I found that linear kernel performs the best. Therefore, I chose linear kernel for the model. The result is shown as follow:

p	10	20	30	40	50
Accuracy	73.70%	84.34%	84.62%	88.14%	88.72%

Figure 4: SVM

Among all the popular Kernels, linear kernel performs the best since the datasets is linear separable. And the best accuracy is 88.72%.

3.4 SRC

For the SRC, I use MATLAB to implement this algorithm. Since the main idea of this work is to get the sparse vector $x_t = argmin\frac{1}{2}\|y_t - Yx\|_1$. I used the CVX library to do the mathematical problems. And the result is shown as follow:
As we can see from fg.5, the highest accuracy is 94.14%, which is the highest among the all four

p	10	20	30	40	50
SRC	89.48%	91.23%	91.97%	92.36%	94.14%

Figure 5: SRC

algorithms. However, this algorithm takes extremely long time to get the result. Using PCA to do dimensionality reduction may reduce the time consumption, it is still considerable compare to other methods.

3.5 Overall Comparison

For this part, the eigenfaces model I used as comparison is the one that deleting 3 highest OC, since it performs better. I plotted the accuracy vs number of trainings samples curves overall comparison among all four algorithms.

Figure 6: Overall Comparison

p	10	20	30	40	50
Eigenfaces	71.92%	80.77%	83.44%	81.54%	83.07%
Fisherface	79.01%	85.55%	80.85%	91.94%	93.91%
SVM	73.70%	84.34%	84.62%	88.14%	88.72%
SRC	89.48%	91.23%	91.97%	92.36%	94.14%

Figure 7: Overall Comparison

From the above comparison we can see that all methods perform well if presented with an image in the test set. And removing the initial three principal components improves the performance of the eigenface method in the presence of lighting variation. The Fisherface methods is the best in handling variation in lighting and expressions. Fisherface is better than Eigenface method while requiring less computation time. Among all the four algorithm, SRC achieved the highest accuracy. However, going through the test process, we know that SRC need a lot of calculations which takes extremely long time. Using PCA to do dimensionality reduction may reduce the time consumption, it is still considerable compare to other methods.

4 Conclusion

In this paper, I introduced four face recognition method, which are Eigenface, Fisherface, SVM and SRC. Then I shows the implementations and results of the models I built for the four methods. The dataset I use is Extended YaleB dataset. I also did the overall performance comparison among the four model. The final results shows that SRC achieved the highest accuracy. However, it takes extremely long time. Fasherface achieved a relative better performance while requiring less computation time.

References

1. P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, ”Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, 711-720, July 1997.

2. C. J. C. Burges, ”A Tutorial on Support Vector Machines for Pattern Recognition”, Data Mining Knowledge Discovery, vol. 2, no. 2, pp. 121-167, June 1998.

3. J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, Robust Face Recognition via Sparse Representation, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 2, pp. 210-227, Feb. 2009.

4. Fasherfaces: http://www.scholarpedia.org/article/Fisherfaces

5. Support Vector Machine: https://en.wikipedia.org/wiki/Supportvector-machine