Viktoriia Varodi
Statistical Learning

EIGENFACES AND FISHERFACES FOR FACE RECOGNITION

1. Introduction

Face Recognition has been under study by data scientists since the beginning of machine learning. An increasing interest exists in identifying or verifying a person because of its variety of applications in different fields: security, surveillance, marketing, entertainment, and human-computer interaction. We can already see some of its uses in our life, such as social media tagging people's faces in images or some mobile phones that only can be unlocked with their owner's face.

For this project, an image dataset was provided in order to perform a classification task by building a statistical model. This face recognition model must be able to tell us if a given image belongs to this dataset or not. If the face image is a member of the dataset, the model must return the corresponding label associated with that face. On the other hand, a 0 is returned when the image does not belong to any of our individuals, we have implemented thresholds to recognize this impostor.

The statistical model was build using different techniques:

- Principal Component Analysis (PCA)
- K-Nearest Neighbours (KNN)
- Fisher Discriminant Analysis (FDA)

Combining algorithms, two different models were obtained:

1. PCA + KNN
2. PCA + FDA + KNN

In both models, cross validation and parameters optimization were performed.

2. Procedure

2.1. Exploratory Analysis

First of all, we need to get familiar with the image dataset. By looking at the images we can see the dimensionality and number of faces.

The dataset contains 150 images of 200x180 pixels per picture. There are 25 different individuals, which means that each person has 6 faces representation.

Each image is labeled with an etiquette as the next example: "10AT.jpg", where the number is the identifier of the human. This identifier is the one we are going to return in our model in case a new picture of them is provided to be recognized.

Picture of the 25 faces belonging to the provided dataset.

2.2. Data Cleaning

Average face

We cannot work with these picture files directly, but we know that images are composed of pixels, which are just numbers in a computer. To work with it, the images must be transformed in vectors in order to manage these numbers. These vectors of 200x180x3 are going to be stored in a matrix data of 150x180.000 dimension. The data is labeled with the corresponding identifier keeping only the number.

For scaling and centering the data, the mean face was calculated which was later subtracted from each of the images in the dataset. At the left we can see how this average face of our dataset looks like.

2.3. Techniques

2.3.1 PCA

Image files are extremely high-dimensional, 200x180 pixel equals 36.000, multiplied by each color (red, green, blue) a total of 180.000 columns per image are obtained. Working with them is slow and we use lots of storage. But very few of the vectors are valid face images. We want to effectively reduce the dimensionality. The dimensionality of the eigenspace is reduced using Principal Component Analysis.

This unsupervised method is computed by finding the eigenvalues and eigenvectors (eigenspaces) corresponding to different cumulative variances. Eigenfaces are the principal components of a distribution of faces, which means, the eigenvectors of the covariance matrix of our training data. These Eigenfaces can be plotted, by plotting the resulting eigenvectors.

It is important to take into account that the dispersion matrix cannot be calculated due to computational problems. To avoid this problem the eigenvalues of Sigma_ were calculated.

Our PCA function takes as input a set of observations and returns an object with the calculated data: matrix containing the eigenvectors (PCA$P_EigenVectors), a vector containing the variance explained by each principal axis (PCA$D_VarianceExplained) and the mean of the input data. We wanted the principal components to represent at least 95% of the original dataset. That is why we write this as a default condition in the second parameter. To choose the parameter for the cumulative variance we test from 0.95 to 0.99.

Once PCA is computed, the new data is calculated by projecting the original data into the new eigenspace.

2.3.2 KNN

Face recognition is a classification task, a supervised method is needed. KNN is the chosen method as the K-Nearest Neighbours algorithm is used to recognize faces by the nearest neighbor.

Our final goal is to see how much different is the new input image with the data that we already have processed, stored and trained. This is done by identifying if the new data belongs or not to the training data and classifying it to the corresponding individual identifier. For this, we want to compute the distances between two observations in the eigenspace.

Our KNN function is going to return the corresponding identifier label or a 0 for impostors. To obtain this 0 we added a threshold value. This threshold is chosen by trial and error. In the way it is implemented, this threshold is going to vary for each test input. This value is calculated by multiplying the maximum distance by 0.15, 0.2, 0,3 or in-betweens values. Then if our minimal distances exceed the threshold, we have found an impostor.

2.3.3. Fisher

Fisher Discriminant Analysis is used to maximize the distance of each class of images at the same time. We have 25 classes corresponding to each person pertenecenting to the Training dataset.

Inside the Fisher function implementation, first we calculate the mean (means) of each class and the mean of the entire train set (mean.train). Then, we calculate the between-class matrix Sb and the within-class matrix Sw, both 24x24 dimension. These matrices are used to get the eigenvectors, the ones we are going to project onto the new train and the new test that previously have been reduced using PCA and keeping 24 principal components.

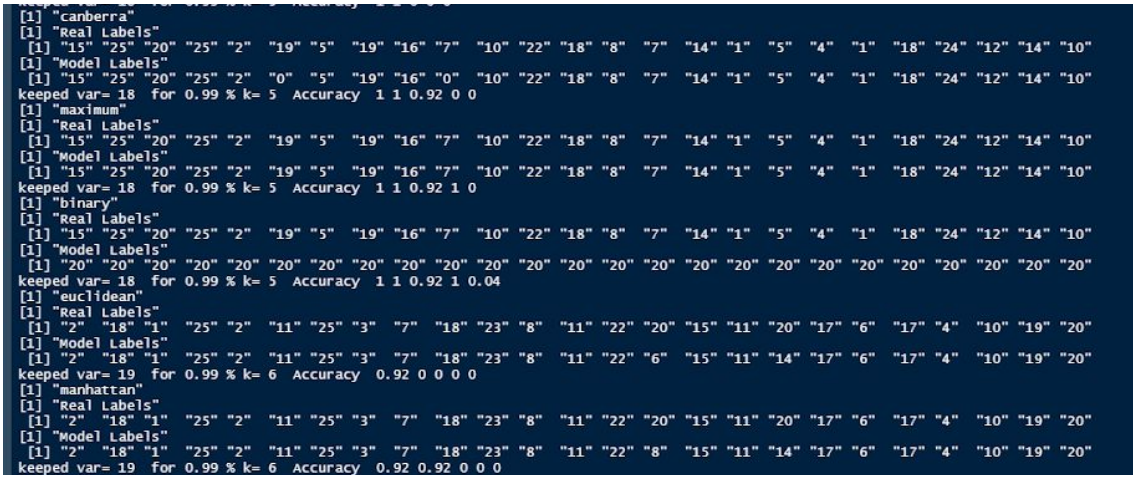
At the beginning, we have keeped 24 fisher eigenvectors. Later, trying to improve the accuracy we decided to follow the same procedure to get the higher representation depending on the input parameter of the variance explained. This way, depending on the keeped variance, the new projection train and set is going to be of 24x13 or 24x21, it may vary. This is something that we are going to see later after optimizing the best parameters for our final model.

2.4. Cross validation and Parameters Selection

We do cross validation of k-folds with 6 folds to obtain a more accurate accuracy for every variance-distance pair. Before that, our rows are randomly replaced to avoid having the same person faces data near each other. The next step is split the data into train and test, to avoid overfitting. Once we have trained the model, we compare the model results with the inputted data labels, so we can calculate the accuracy. This process is repeated 6 times with different combinations of parameters:

- k = 1, 2, 3, 4, 5, 6
- distance = “euclidean”, “manhattan”, “canberra”, “maximum”, “minkowski”
- variance = 0.95, 0.96, 0.97, 0.98, 0.99

In the for loops each time we are testing different sets we print in the screen the “Real Labels” and the “Model Labels”, to see at real time how our model is making the classification and the corresponding accuracy, this way we can see easely which row is failing to classify and differentiate if real labels are setting as impostors or vice versa.



3. Results

Both combinations are obtaining really high accuracy, which seems to be a good indicator but at the same time it makes it difficult for us to distinguish which are the better distance-variance combinations to choose for our final model.

```
> save_results
[[1]]
  euclidean manhattan canberra maximum binary
1         1         1        1    1.00   0.04
2         1         1        1    1.00   0.00
3         1         1        1    1.00   0.00
4         1         1        1    0.84   0.00
5         1         1        1    0.96   0.04
6         1         1        1    0.96   0.12

[[2]]
  euclidean manhattan canberra maximum binary
1         1         1     1.00    1.00   0.04
2         1         1     1.00    1.00   0.00
3         1         1     1.00    1.00   0.00
4         1         1     1.00    0.84   0.00
5         1         1     1.00    0.96   0.04
6         1         1     0.96    0.96   0.12

[[3]]
  euclidean manhattan canberra maximum binary
1         1         1        1    1.00   0.04
2         1         1        1    1.00   0.00
3         1         1        1    1.00   0.00
4         1         1        1    0.84   0.00
5         1         1        1    0.96   0.04
6         1         1        1    0.96   0.12

[[4]]
  euclidean manhattan canberra maximum binary
1         1         1     1.00    1.00   0.04
2         1         1     1.00    1.00   0.00
3         1         1     0.96    1.00   0.00
4         1         1     1.00    0.84   0.00
5         1         1     0.88    0.96   0.04
6         1         1     1.00    0.96   0.12

[[5]]
  euclidean manhattan canberra maximum binary
1         1         1        1    1.00   0.04
2         1         1        1    1.00   0.00
3         1         1        1    1.00   0.00
4         1         1        1    0.84   0.00
5         1         1        1    0.96   0.04
6         1         1        1    0.96   0.12
```

1. PCA + KNN

```
> save_results
[[1]]
  euclidean manhattan canberra maximum binary
1         1      1.00     0.96       1   0.04
2         1      1.00     0.84       1   0.00
3         1      1.00     0.92       1   0.00
4         1      0.96     0.84       1   0.00
5         1      1.00     0.88       1   0.04
6         1      1.00     0.88       1   0.12

[[2]]
  euclidean manhattan canberra maximum binary
1         1      1.00     0.92       1   0.04
2         1      1.00     0.84       1   0.00
3         1      1.00     0.92       1   0.00
4         1      0.96     0.84       1   0.00
5         1      1.00     0.88       1   0.04
6         1      1.00     0.80       1   0.12

[[3]]
  euclidean manhattan canberra maximum binary
1         1      1.00     0.88       1   0.04
2         1      1.00     0.80       1   0.00
3         1      1.00     0.96       1   0.00
4         1      0.96     0.80       1   0.00
5         1      1.00     0.88       1   0.04
6         1      1.00     0.80       1   0.12

[[4]]
  euclidean manhattan canberra maximum binary
1      1.00      1.00     0.88       1   0.04
2      1.00      1.00     0.84       1   0.00
3      1.00      1.00     0.96       1   0.00
4      0.96      0.96     0.80       1   0.00
5      1.00      1.00     0.88       1   0.04
6      1.00      1.00     0.76       1   0.12

[[5]]
  euclidean manhattan canberra maximum binary
1      1.00      1.00     0.88       1   0.04
2      1.00      1.00     0.80       1   0.00
3      1.00      1.00     0.88       1   0.00
4      0.96      0.96     0.80       1   0.00
5      1.00      1.00     0.88       1   0.04
6      1.00      1.00     0.80       1   0.12
```

2. PCA + FDA + KNN

Matrix 1 at the top corresponds to 0.95, 2 to 0.96, and so on until the bottom with matrix 5 for 0.99. Binary distance is set as another parameter even though we know is not a valid distance for our data, it is set just to make a visual signal.

In the first model with PCA + KNN we chose euclidean distance, k = 3 and 0.95 as optimal parameters.

In the second model with PCA + FDA + KNN we chose manhattan distance, k = 1 and 0.95 variance, which means we have keeped 13 fisher eigenvectors. At the matrix it seems as the Maximum distance is always getting 100% accuracy, but after building our model function and playing with the data leaving out impostors, the way we have implemented the threshold does not work with this distance. Then Euclidean and Manhattan were left, both performing excellently so the decision was made frivolously. Same for choosing between 0.95, 0.96 and 0.97 variance.

We are going to use these parameters also to build our Eigenfaces and our new fisher data set. This are the 24 Eigenfaces our final training data has:

This are the 13 Fisherfaces:

Once we have saved the data using those parameters, with the training fisher set and other utilities (utilsFisher.RData) the face recognition function model is done.

4. Conclusion

The objective of our task is achieved. Both algorithms combinations give us really good results. The main difference is really small, and it was difficult to see it and to test these extreme cases when we are always obtaining a good match. The difference comes with the impostors. Our PCA + FDA + KNN model is more precise and does not make any false assignment, while PCA + KNN can assign a face as belonging to our dataset when it does not. This is why we have decided to keep PCA + KNN + FDA as our final model.