**APPLIED DATA SCIENCE FINAL PROJECT**

**FACIAL KEYPOINTS DETECTION**

Yuezhi Wang (yw2586)   Wenkai Pan (wp2191)

Min Shi (ms4786)    Mi Xiong (mx2152)

**Abstract**

Facial keypoints such as the center of eyes and nose tips are important features for many different tasks in automatic face processing. The detection and localization of facial keypoints is usually performed interactively or it is not very precise. The goals of our project are to develop an accurate model to accurately measure facial features, tracking characteristic edge and output key points of the facial objects to be searched. The whole searching and analyzing process of our project is based on a dataset from a Kaggle competition and is principally inclined to use general methods like Trees, Regularization regression and Principal Components Analysis (PCA).

**Keywords:** Facial keypoints detection and localization, face recognition, trees, regularization, principal components analysis

1. **Introduction**

Facial keypoints detection is a computer technology that finds the locations and sizes of human face main organs like mouth, nose and eyes in an image or a sequence of images. As one of the most important problems to be solved in human face processing, locating and extracting the key points from a whole human face with and without background is currently a very active research due to its potential applications like automatic face feature recognition, security system, video conferencing, intelligent human-computer interaction. During last decade, a large amount of methods have been proposed for face detection and facial keypoints detection. These methods can be generally divided into two categories: feature based approaches and image-based approaches (or named classification-based approaches).

The goal of our project is to detect the location of both eyes and nose on a human face based on the information of a grey scale picture.

1. **Data Processing**

In this project, we mainly use dataset gained from Kaggle[i]. Since it is an open competition, the result of the test data is not given. So we split the training data into two parts, and the second 1000 observations are used as test data. Our goal is to detect the locations of eyes and nose.

**2.1 Training Data**

Training data contains 6049 training images. And for each image there are image ID and 6 variables corresponding to the target facial keypoints’ x and y coordinates: Left eye center, Right eye center, and Nose tip. The image variable ‘image’ contains 9216 Grey Values from 0 to 255, which can output a 96\*96 pixels image with key points marked (figure 2.1).

**2.2 Testing Data**

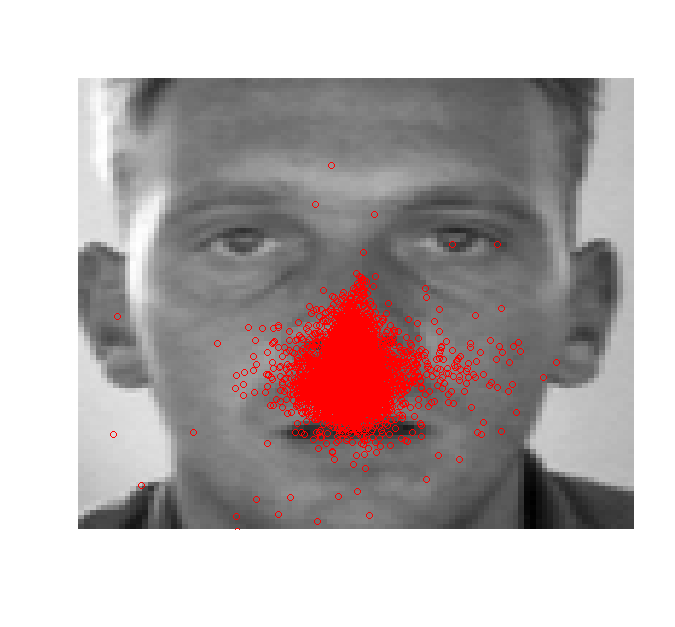
Test data contains 1000 test images with image IDs and ‘image’ variable with 9216 Grey Values from 0 to 255, which output a 96\*96 image (figure 2.2).



*(figure 2.1 Training data image with keypoint)          (figure 2.2 Test data image)*

**2.3 Exploratory analysis of the data**

Based on common sense, human faces are more or less similar. If the pictures in training data are presented as the one shown above, it is simpler to locate the feature. So we need to see how variable our data is. One good way is to plot out the locations of all the nose tips (figure 2.3).



*(Figure 2.3 The location of nose in the training data)            (Figure 2.4 One of the outliers)*

Most nose points are located in the center of an image as we expected, but there are certain outliers that deserves further investigation, as they could be errors. One extreme outlier is shown in figure 2.4, which clearly is not an error. This indicates we cannot treat outliers as normal data, as the background information beyond a face could cause inaccuracy. So we need to eliminate the effect of the noise information as much as we can.

**2.4 Noise Elimination**

As we shown above, the main problem of local and data-driven keypoints detection is that there is too much local image structure in complex real world images. Therefore, it is careless to give any interpretation to local features without considering their context. Thus, our approach starts by eliminating background noise.

<Concepts and steps of noise elimination>

<Result>

1. **Methodology**

We have come up with several methods to extract the features. We use the accuracy of a very simple method as the baseline to assess the models we built. The baseline method (Naive Mean Method) is simplistic as it utilized a quite general method which didn’t take into consideration of individual difference. The test error of the Naïve Mean Method is

**3.1 Key point Box matching method**

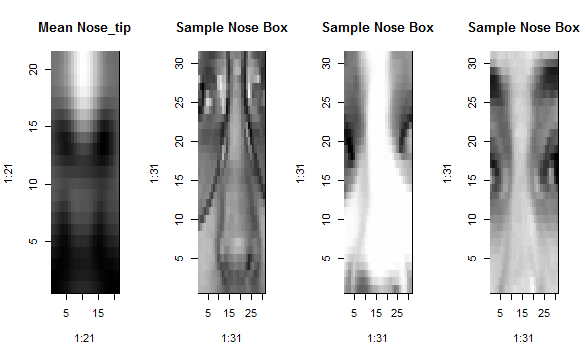
Assumption: In predicting problem for one specific key point, we should highlight those pixels that are around that key point.

Motivated by this assumption, we design our new strategy of using “Image Box”: Instead of utilizing all information of a picture, we use small box to extract out matching area in picture for one specific key point. This method is a two-step strategy: first calculate a mean key point and pick the best match of the mean key point to all possible small boxes in one test data.

To further illustrate this method, we first focus on one single key point, i.e. nose tip.

For each picture in training dataset, we extract a 21\*21 (after trial and error) square box around nose tip. After we got all 6049 21\*21 boxes, we calculate the average of all these 6049 matrices.

The result is presented below.



Now we can use this Mean Nose tip box to “search” for the same keypoint in the test images.

Given a test image we need to try all its possible 21\*21 boxes, and see which one best matches the mean-nose box. Our measure to determine the best match is correlation with the mean-nose box. The coordinate with the highest score of correlation with that mean-nose box is considered to be our predicted location of nose.

We can easily apply the same idea to other key points.

Compared with Naive Mean method baseline, this feature box extraction method achieves a prediction error score of 3.702. It’s an improvement and the result makes sense.

**3.2 Combine PCA with KNN**

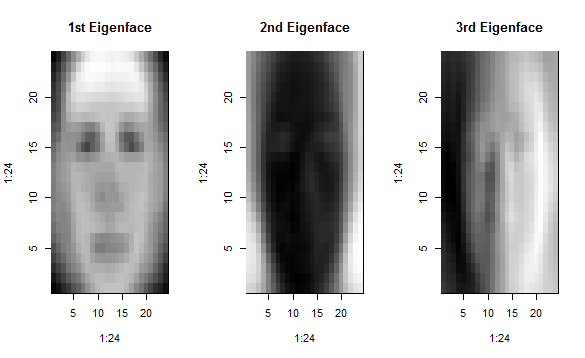
Assumption: If two pictures are “close” enough, then the location of key points of the two faces are very similar. Inspired by this idea, we would like to detect the “nearest neighbor” of each test picture in the training datasets, and immediately assign the coordinates of that nearest training picture to the test picture.

However, it is hard to measure the distance. If we choose Euclid Distance measure, there still exists severe problem about “huge” data. Here we have altogether 7049 pictures, each represented by a 96\*96 matrix (or a vector with length 9216). To calculate distance matrix for this dataset with such huge number of dimensions is a disaster.

Dimension reduction is something covered in basic machine learning topic. We immediately turn to PCA (Principle Component Analysis)technique.

To do PCA on a 7049\*(96\*96) matrix still suffer typical “n<<p” problem. To solve this, we apply a simple strategy to compress each picture from 96\*96 to 24\*24. We take the average of each 4\*4 box in original picture and replace initial 16 pixels with one value. This method ends up with a very satisfying result. After all, we are trying to reduce dimension from 96\*96 to something like 30 or less, this initial step with 24\*24 is reasonable.

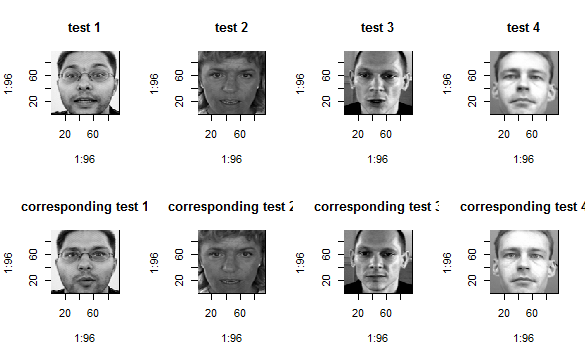
The next step is to implement PCA on training set and use the first 30 scores as a vector to represent data. It includes creating the mean face from the training set, subtracting the mean face, and running *prcomp()* in R on the resulting image matrix. Project the testing data onto the first 30 loadings so that it is also represented by the first 30 scores. The following is the “Eigenfaces” (eigenvalues).

****

*Figure 3.2: The first three “eigenfaces” of the data*

Next step is to use 1-NN classification in the space of the first 30 scores to identify the subject for each testing observation.

Let’s look at some of the matched pairs given by our PCA&1NN method.

****

Visualization indicates that our combined method works perfectly as it can exactly picks out the same person in training data corresponding to test data! We have gained some very interesting insight about this dataset: This dataset might be originally from so-called “facial detection” problem, which is interested in identifying the same person’s picture in training data. We can definitely say that our method did a very good job in this problem. The difference of two pictures are merely slight different pose of that person or different angle of camera. It seems like magic and we are very confident that PCA retain very useful information of the initial data so that the two corresponding pictures are really close enough.

However, our main concern lies in whether this method has good performance of our interested question in terms of prediction error? Unfortunately, the answer is NO. Our test error has a bad performance of 5.654. However, we also notice that the error in the first 4 test data is 1.531, 1.370, 1.684 and 1.193 which is quite satisfying not only compared with the error in this method but also with baseline. This gives us the message that our method has a good performance on “facial detection” problem. If the same person is detected, the prediction error of key point detection is satisfying. If there is no such same person matched in training data, it seems to be not that reasonable to utilize the nearest neighbor to give key points’ location. The reason is that PCA still fetch out those general feature of the whole picture. In order to focus our attention on “key points”, we still need to extract the key feature of that specific box of pixels. This is the motivation of our following method, which applies PCA on key points area.

**3.3**

**3.4 Mi Xiong**

**4  Result**

**4.1**

**4.2**

**4.3**

**4.4**