ISyE 6416 – Basic Statistical Methods - Fall 2015
Bonus Project: “Big” Data Analytics
Final Report

Team Member Names: Hugo Acuna, James Netter, Mahadevan Vaidyanathan

Project Title: Facial Recognition

Problem Statement

From social networking sites to law enforcement, many different organizations around the world are adopting tools and software that allow them to recognize a person’s face. Facial recognition is important for these groups as they can be used for security purposes, network building, and even geolocation efforts. There are, however, obstacles in the way of creating such a system. The ways a face can be presented to a camera (ie. black and white vs color, shading on the face), or lack of past facial data to compare a new face with. Basically, you are not guaranteed to have every face flowing through a database to be presented to the detecting software in the same way. To combat this, certain methods have thus been adopted with the hope of mitigating some of these factors. Our team decided to work on a method that would be able to detect and output key features from a person’s face. However, there are many ways that could be used, so we attempted to find the best method for this.

Data Source

Using data from an online source, we found a dataset of faces which we used to test our methods. The dataset contains two-dimensional, 8-bit, grayscale facial images with corresponding facial points listed in typical (x,y) coordinate format. Each image is represented by a 96 x 96 pixel matrix, whose entries are integers from 0 to 255. These integral values are meant to characterize the intensity of each pixel, for a total of 9216 pixels in each image. The dataset is a matrix of size 7049 x 31, where each row is made up of one facial image, out of 7049, with 30 columns made up of the 15 facial points’ (x,y) coordinates, and finally the 9216 numbers representing the pixel matrix of each image melted row by row. The 15 facial points marked in each image correspond to nose, eye, mouth, and eyebrow locations on the face.

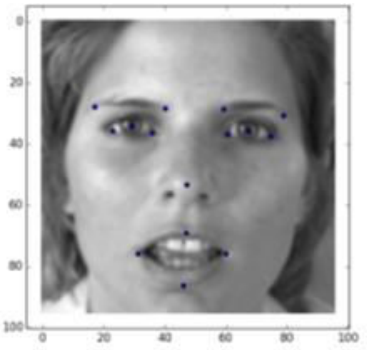


Image 1: Example of image from dataset. Key facial features marked

Data source: https://www.kaggle.com/c/facial-keypoints-detection/data ,

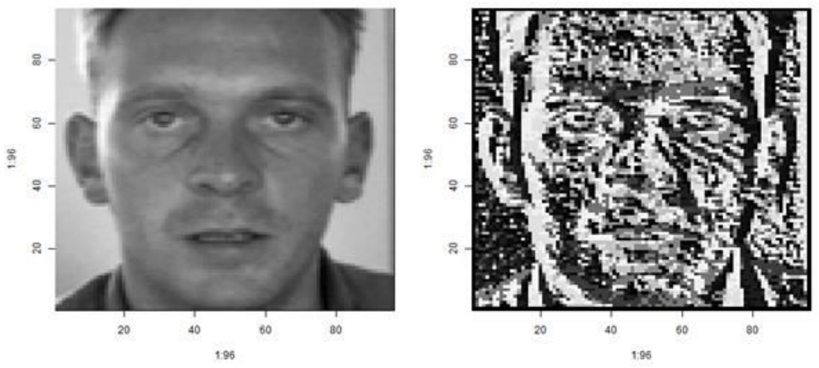
Methodology

As mentioned previously, each image is represented as a 96x96 pixel matrix, and for the data used to train our solutions, fifteen key facial features for each corresponding picture are marked in the dataset matrix. Our methodology thus entailed taking the given points for each face and using them to predict where a new image's points would be. We use hold-out cross-validation with 80% as training dataset and 20% as testing dataset.

Pre-processing methods

We used several different methods to analyze and prepare the pictures for further analysis. One such way was through local binary patterns, which converted pixels into a binary variable based around a center point. This reduced the sensitivity of an image to illumination from any light source. Since each picture can be represented through matrices, and each matrix from the photo dataset's vector values are indicative of pixel intensity, LBP can be done as follows

1. Choose a center point
2. If a surrounding data point is higher, change value to 1, otherwise change to 0
3. Transform the binary sequence to decimal
4. Output new image



Images 2,3: Before, after picture of LBP analysis

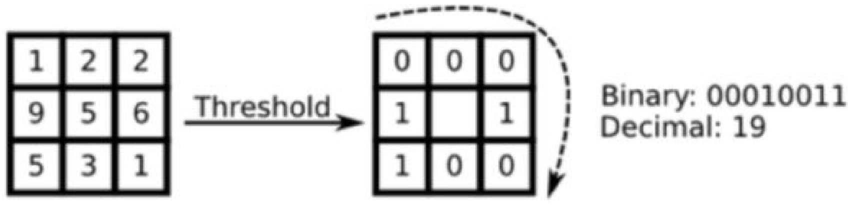


Figure 1: Representation of LBP analysis

We also performed Principal Component Analysis on each picture, with the following steps.

1. Obtain face images $I_1, I_2, \ldots, I_m$
2. Represent every image I_i as a N^2 * I vector Γ_i
3. Compute the average face vector $\Psi = 1/M \sum \Gamma_i$

4. Subtract the mean face: $\Phi_i = \Gamma_i - \Psi$
5. Compute the covariance matrix $C = 1/M \sum \Phi_i * \Phi\hat{}T = AA\hat{}T$
6. Calculate the ordered sequence of eigenvectors (eigenfaces) and eigenvalues of covariance matrix
7. Project dataset over the 200 principal eigenfaces

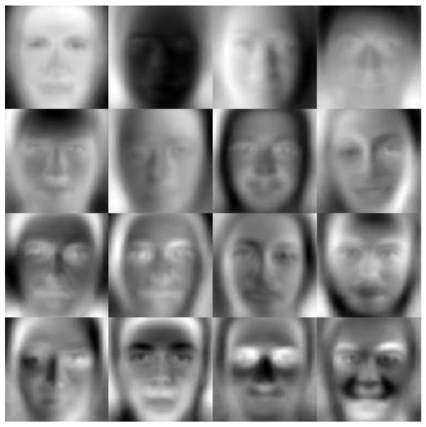


Image 4: First 16 eigenfaces

From the figure below, it can be seen that the first 200 eigenfaces or components explain about 93% of the data variance.

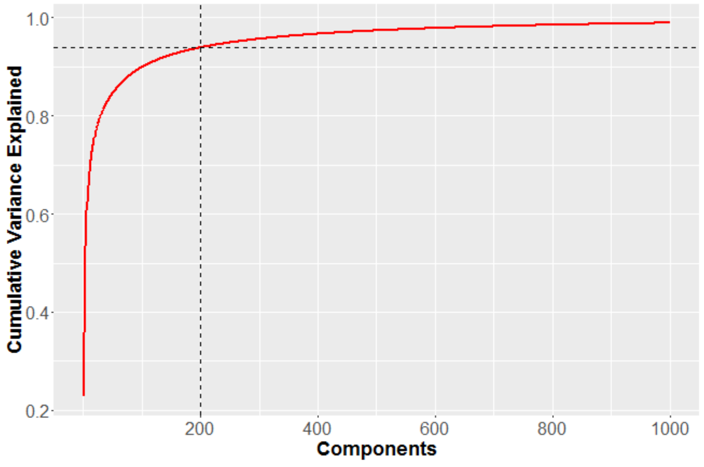


Figure 2: Cumulative variance vs number of principal components considered

The image below shows a reconstructed face using the first 200 eigenfaces.

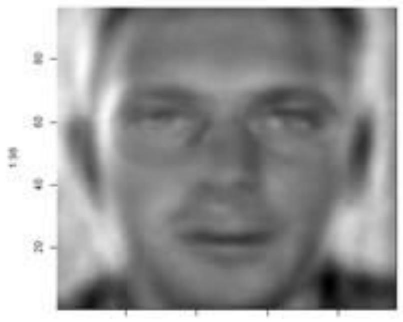


Image 5: Face projected over the 200 principal components

By using these methods to further process each picture, we made it easier to identify more prominent features in a face when we implement supervised learning methods.

<ins>Supervised learning methods</ins>

After preprocessing the data, we applied two supervised learning methods in order to identify facial features: Mean Patch Searching (MPS) and Linear Regression (LR).

Mean patch searching involves taking windows of certain size around the training (x,y) of a facial feature for all training images, and then taking the average obtaining the mean patch of the facial feature. After that, a window of the same size is superimposed over each testing image in different positions, searching around the mean (x,y) of the requested facial feature the set that has a max correlation within the mean patch. The positon of that window is the predicted (x,y) of the facial feature in a testing image.

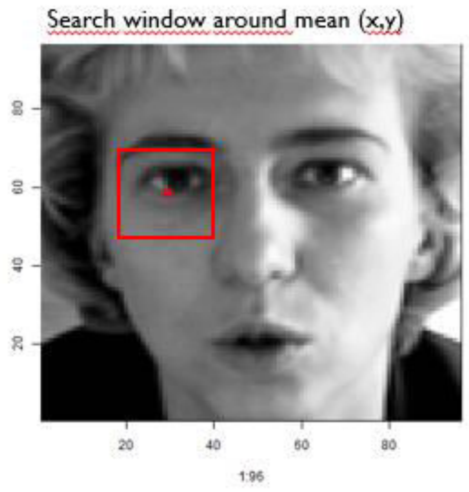


Image 6: Mean Eye facial feature with 21x21 window

Linear regression involves taking all pixels inside a window of certain size around the mean (x,y) of a facial feature as explanatory variables, and the position (x,y) of the facial feature as a response variable.

We then did this for all faces, with the three different types of preprocessed pictures, to find the associated mean square error of each, the smallest of which will be our selected method.

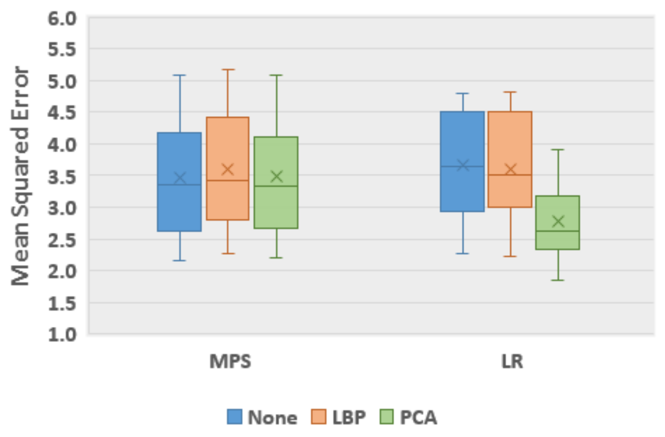
<ins>Evaluation and Final Results</ins>

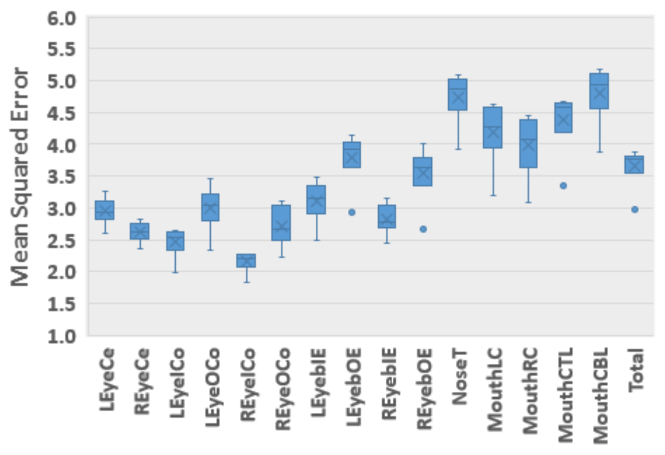
In this section we show the results in terms of the individual and total Mean Squared Error (MSE) of key facial feature coordinates in pixel units. We obtained 90 values of individual errors corresponding to the combination of 3 preprocessing methods, 2 supervised learning methods and 15 key facial features. These values are displayed using boxplots in order to compare methods and facial features.

Figure 3 below compares the different combinations of preprocessing and supervised learning methods. Label "None" was assigned to results without using pre-processing method.

We can see a significant improvement when PCA is applied to linear regression. On the other hand, there is no improvement when data is pre-processed using Local Binary Patterns.

Figure 4 compares the different key facial features. Here we can see that eye and eyebrow features (left boxes) have lower errors than nose and mouth features (right boxes). Also we can see that features located at the right side of the face (REye, REyeb, MouthRC) have lower errors than their corresponding left sided features (LEye, LEyeb, MouthLC).





Figures 3, 4: Boxplots comparing methods and key facial features in terms of individual feature MSE

The table below shows the total Mean Squared Error for each combination of pre-processing and supervised learning methods. This confirms that the best performance was achieved by PCA+ Linear Regression method.

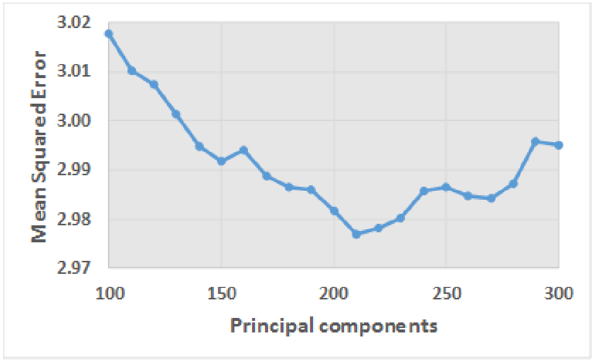
Table 1: Table comparing methods in terms of total MSE

Preprocessing methods	**Supervised Learning Methods**	
	Mean Patch Searching	**Linear Regression**
Unaltered Image	3.74	3.80
LBP altered Image	3.88	3.78
PCA altered Image	3.76	**2.98**

The previous results were obtained using a square window of side 21 pixels and 200 principal components when PCA was used as a pre-processing method. We investigated the effect of change these two parameters for the PCA + Linear Regression method.

Figure 5 shows the error for different numbers of components between 100 and 300. Here the minimum error was obtained with around 200 components, hence our previous selection was correct. However, the range of errors is only around 0.05 pixels, thus there is no big difference in choosing anything between 100 and 300 principal components.

Figure 6 shows the error for different window sides. Here the error decreases when the window side increases up to 15 pixels. For windows of side larger than 15 pixels, the error remains constant.



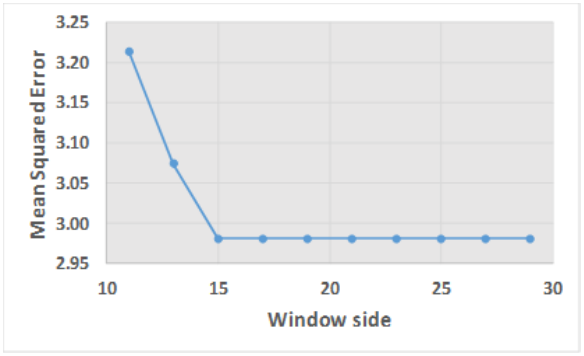


Figure 5,6: Graphs showing optimal number of principal components and window size

From the above figures, it can be seen that when using PCA with linear regression, we will have the best method for finding the key facial features in a picture. We can also see that Local Binary Pattern doesn't really change the effectiveness in any meaningful way. To see the final solution in action, each team member also implemented the analysis with a picture of their own face to see the result.

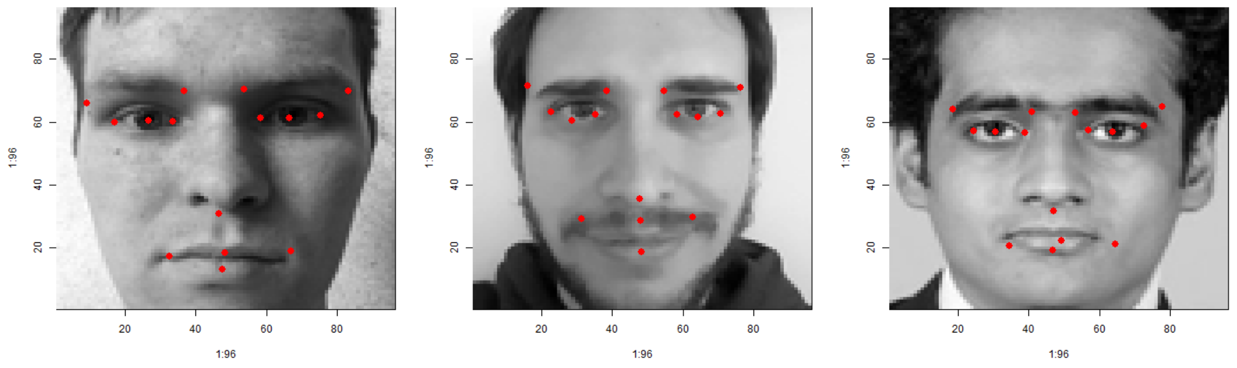


Image 7,8,9: Key Facial Features detected by PCA+LR analysis

Overall, our final method was able to accurately predict facial features within a reasonable margin, and can be used to add to the robustness of other software for facial detection.

In this project, all team members contributed equal amounts to the assignment. Everyone had a say in determining the project scope and locating the data source used. Hugo worked on the code that pre-processed the images, Mahadevan worked on the implementing these processed images with mean patch searching and linear regression, and James performed the mean squared error evaluations of each method, as well as writing up the first drafts of the report. The other members helped edit the final report.