Prediction Model for Human Emotion via Gradient Descent

Tianqi Zhao
PID: 14150941
email: tiz137@ucsd.edu

Siqi Huang
PID:A14143758
email:sih041@ucsd.edu

1. Abstract

In this project, our objective is to use regression to separate faces of different emotions. We use the dataset from the California Facial Expressions (CAFE). We approach our goal through two regressions, logistic regression via gradient descent and implementing softmax regression via gradient descent

2. Load and Preprocess the Data

2.1 Why is it important to only use the training data to compute the principal components?

It is important to only use the training data to compute the principal components because we do not want to overfit on our dataset. We split our dataset into training, testing and holdout in order to check how the algorithm we train would work on this set of data. If we include the testing and holdout data when computing the principal components, we might have perfectly fit to the whole dataset but also would lead to overfitting.

2.2 Display of six emotions

The display will automatically pop out while running our code.

2.3 Eigenface display

The display will show up as the faces are stored as "pca_display.png". (Figure 2.3)

3. Logistic Regression via Gradient Descent

3.1 Evaluation of Separation between Happy and Maudlin Faces

i) The four graphs over the average and standard deviation for test and holdout errors for five by transforming images into 1, 2, 4 and 8 principal components to train the logistic regression are shown in Figure 3.1, Figure 3.2, Figure 3.3 and Figure 3.4, respectively.

ii) By using the weights corresponding to lowest holdout rate in each run for 1, 2, 4, and 8 principal components, the average test accuracy over 5 runs for the above principal components are as follows:

The test accuracy for one principal component over five runs is 65%(0.1224744871391589); the test accuracy for two principal components over five runs is 70%(0.18708286933869706); the test accuracy for four principal components over five runs is 85%(0.2); the test accuracy for eight principal components over five runs is 95%(0.09999999999999999).

iii) According to the above training, the best number of principal components is 8 since it has the highest average accuracy. In this case, we train our data with 8 principal

components with different learning rates 0.0001, 0.2 and 0.99 to test on the dataset. Figure 3.5, 3.6, and 3.7 corresponds to the different learning rates.

3.2 Evaluation on Afraid and Surprised Faces

The plotting about the average and standard deviation for test and holdout errors for five runs by transforming images into 8 principal components with a learning rate of 0.2 to train the logistic regression is shown in figure 8. The test accuracy for the above training is 75%(0.22360679774997896).

This result is different from when we are doing for happy and maudlin faces. According to figure 3.8, it seems that with such values the loss increases when the number of epochs increase, which is different from when we are doing the separation for happy and maudlin faces. The reason why the learning rate works well on the separation of happy and sad faces but does not work well on this separation might be that these are two different datasets and thus it is possible that it might not work well.

4. Multi-classes Emotion Recognition Softmax Regression via Gradient Descent

4.1 Softmax implementation and evaluation on all six emotions

Here, we chose only the happy faces for each subject and discarded all the happy-with-teeth images. The reason for the selection is that if we have 2 images of happy, and only 1 image of the other emotions, the training result will be based on disproportionate data. So using all of the happy faces will results in skewness of detection toward certain category.

For the training, we figured out that setting learning rate to 0.01 and the number of principal components to 11 will generate the least error in general. The loss on training and loss on holdout sets vs. number of iterations of gradient descents is attached. (Figure 4.1.1) The 6x6 confusion matrix for this data based on the test set results is saved as screenshot and attached. (Figure 4.1.2)

4.2 Comparison over batch versus stochastic gradient descent

Stochastic gradient descent was implemented similarly to the batch gradient descent, with the only addition of reshuffling the elements in the training set before each iteration. The batch and stochastic gradient descent perform impressively similar in such a low number of epochs(20), as shown in Figure 4.2.1.

Stochastic gradient descent should be faster when there are a large amount of dataset and fewer number of epochs. The reason is that by randomly selecting each one data to train, it does not need to go over all the whole data set for one gradient descent. But batch gradient descent goes over the whole dataset for each iteration. In reality, it is very possible that we only need a portion of the data sets to train to have a good model, so using stochastic gradient descent will save a lot of time.

4.3 Visualization of weights

The visualization of the weights obtained by one run of batch gradient descent approach are displayed and saved as “visual_of_’initial of the emotion’.png” file in the directory. The figures are listed in Figure 4.3.1 to 4.3.6.

The reason that the weights and eigenvectors produce such a series of ghost-like images is that since we are multiplying weights with the eigenvectors, we are getting the “average face” and multiplying it with “the weights of emotions”. Therefore, we will get images where each pixel is lightened only if it is lighted in the most images in the same category. In short, the images display the key features that weighs the most in each emotion.

5. Contribution

Tianqi and Siqi discuss together for the implementation of logistic regression and softmax implementation. While writing the codes, Siqi is the main person to implement logistic regression via gradient descent and Tianqi is the main person writing the code for softmax regression via gradient descent. Though we have been responsible for different parts of codes, we share our understandings of gradient descent and figure out the right algorithm of gradient descent.

6. Citations, Figures, Tables, and References

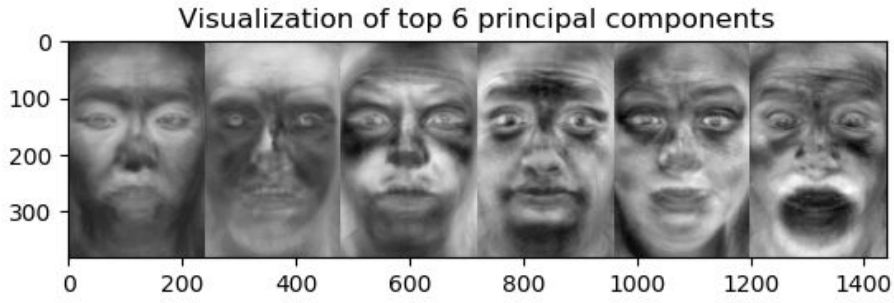

Figure 2.3: display of 6 eigenfaces stored in pca_display.png

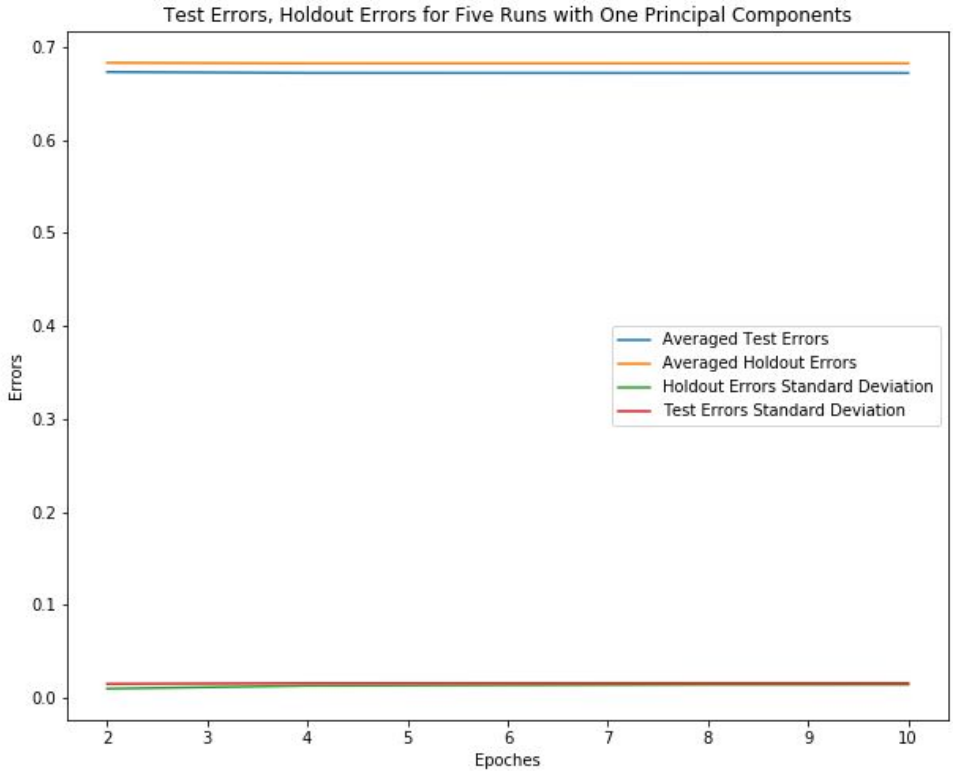

Figure 3.1 Plot of average test errors and holdout errors, and standard deviation of holdout errors and test errors for five runs in the separation of happy and sad faces when the images are transformed to one principal component.

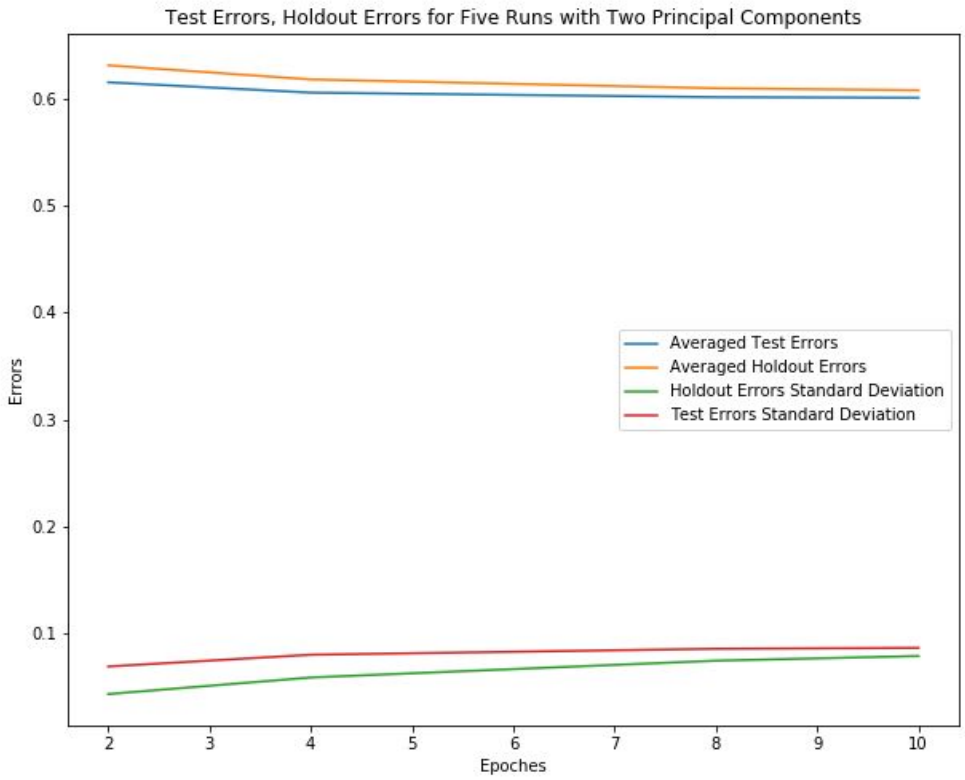

Figure 3.2 Plot of average test errors and holdout errors, and standard deviation of holdout errors and test errors for five runs in the separation of happy and sad faces when the images are transformed into two principal components.

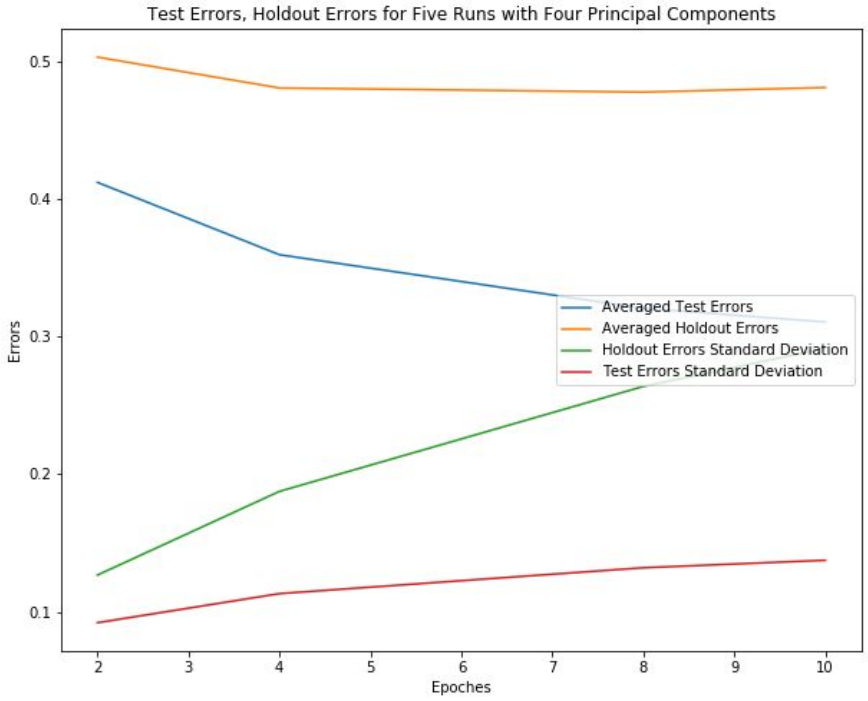

Figure 3.3 Plot of average test errors and holdout errors, and standard deviation of holdout errors and test errors for five runs in the separation of happy and sad faces when the images are transformed into four principal components.

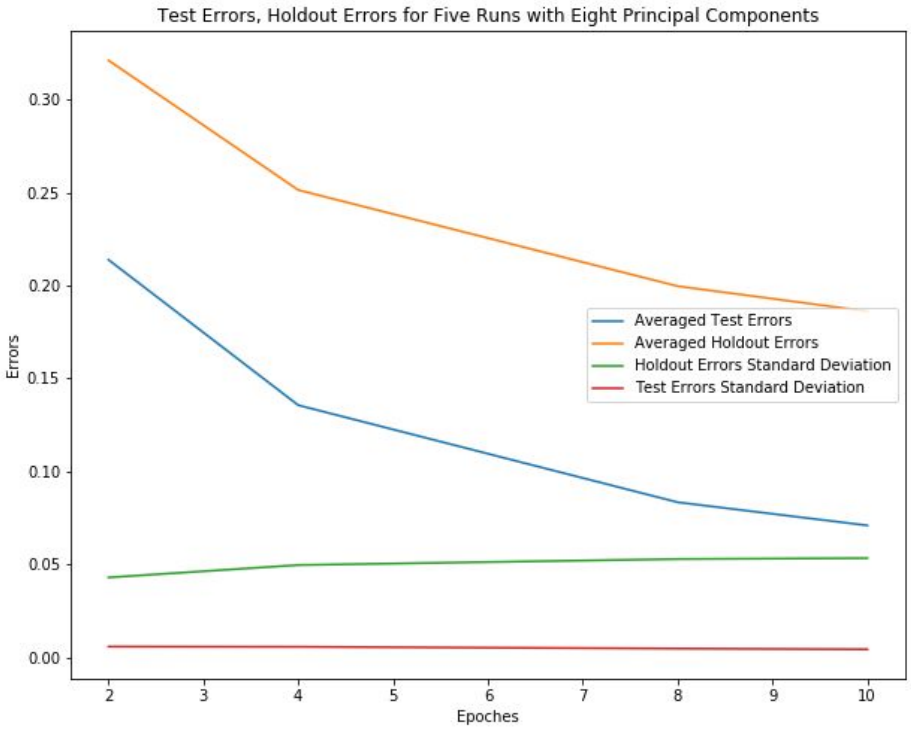

Figure 3.4 Plot of average test errors and holdout errors, and standard deviation of holdout errors and test errors for five runs in the separation of happy and sad faces when the images are transformed into eight principal components.

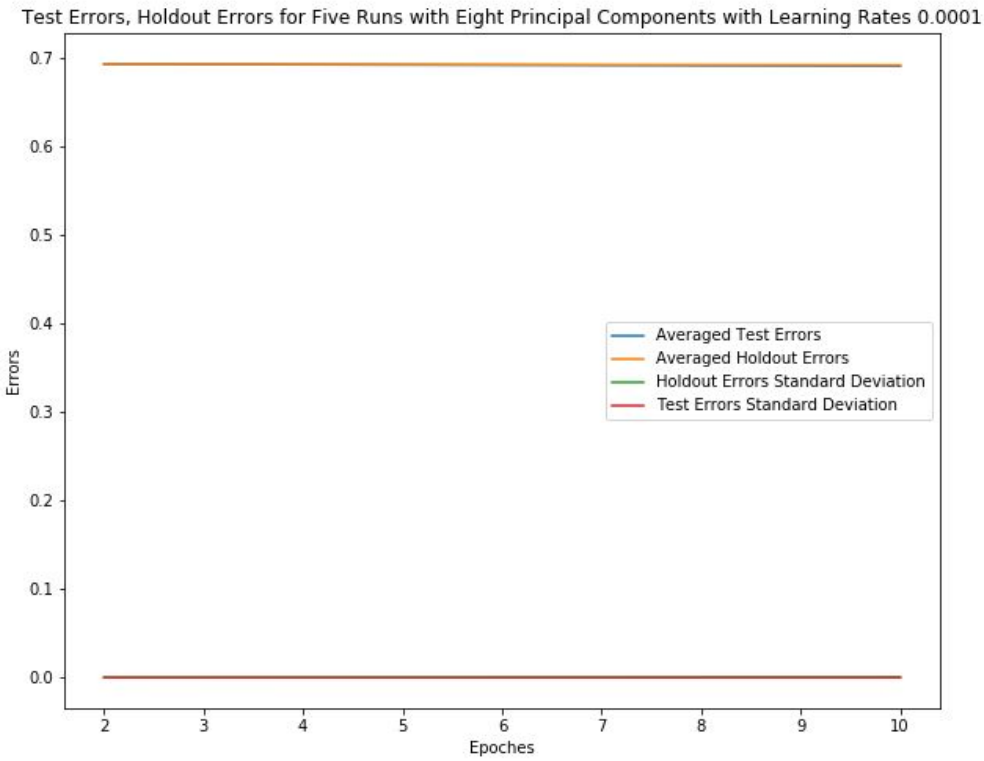

Figure 3.5 Plot of average test errors and holdout errors, and standard deviation of holdout errors and test errors for five runs in the separation of happy and sad faces when the images are transformed into eight principal components with a learning rate of 0.0001.

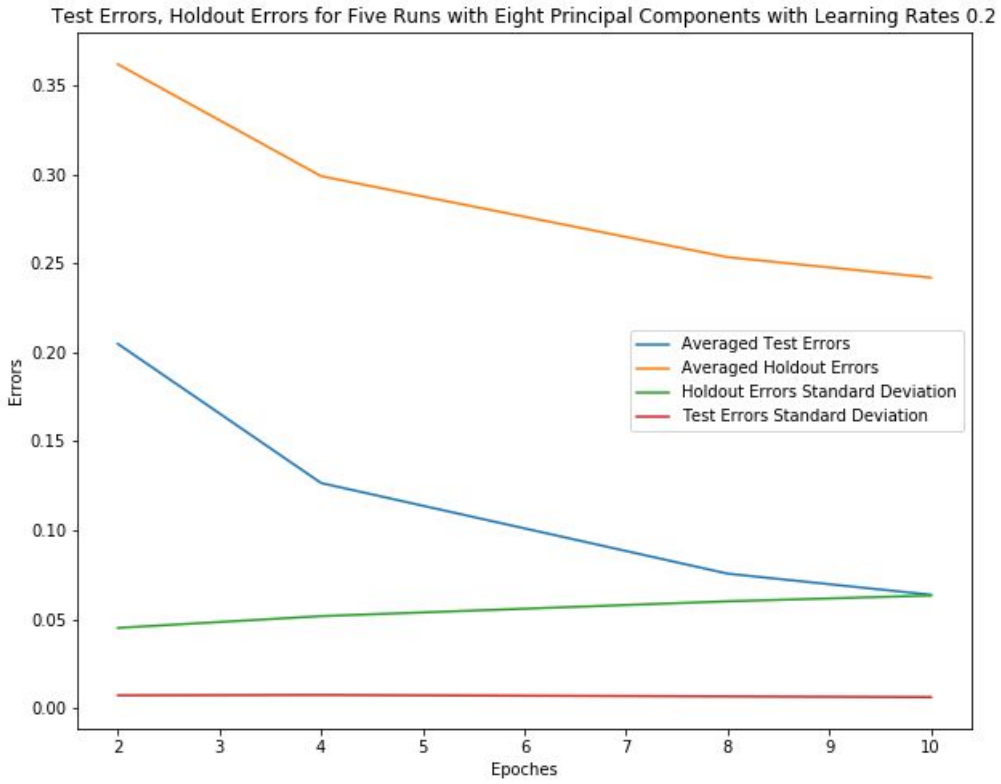

Figure 3.6 Plot of average test errors and holdout errors, and standard deviation of holdout errors and test errors for five runs in the separation of happy and sad faces when the images are transformed into eight principal components with a learning rate of 0.2.

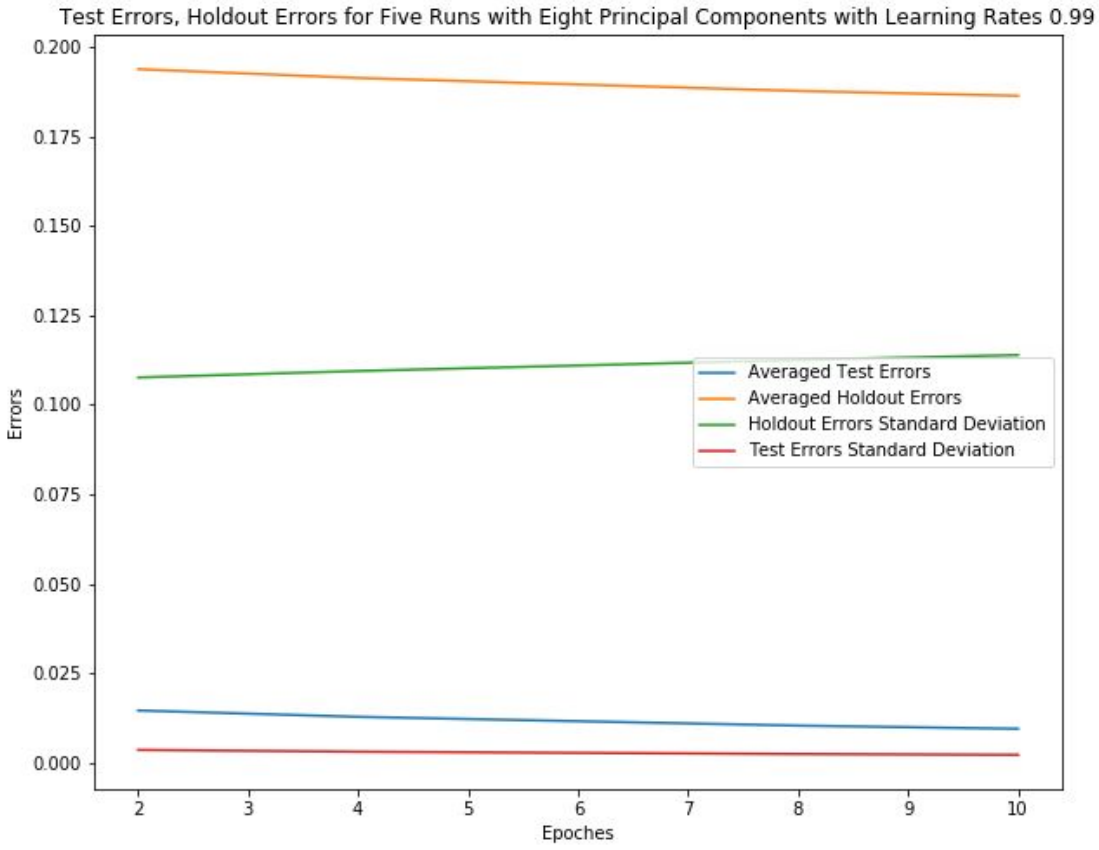

Figure 3.7 Plot of average test errors and holdout errors, and standard deviation of holdout errors and test errors for five runs in the separation of happy and sad faces when the images are transformed into eight principal components with a learning rate of 0.99.

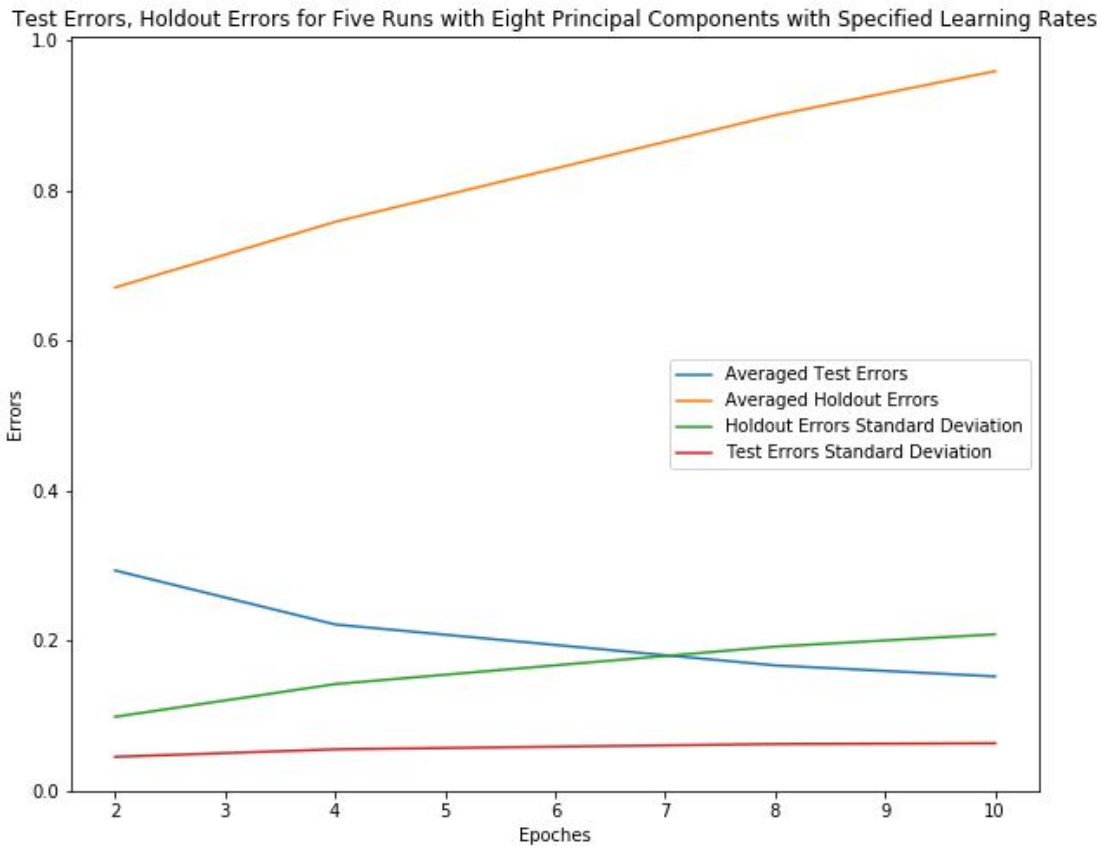

Figure 3.8 Plot of average test errors and holdout errors, and standard deviation of holdout errors and test errors for five runs in the separation of fear and surprised faces when the images are transformed into eight principal components with the specified learning rate of 0.2.

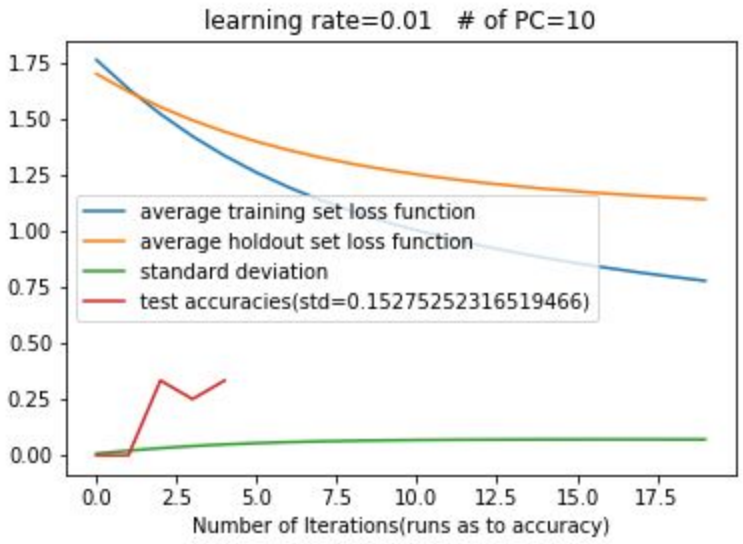

Figure 4.1.1: The plot of the average training loss function and holdout set loss function and standard deviation are saved as 3(a). Notice that the x axis is the time of runs as to test accuracies. One example is shown here.

['h', 'm', 's', 'f', 'a', 'd']
h: [[0.08333333 0.08333333 0. 0. 0. 0.]
m: [0. 0.08333333 0. 0. 0.08333333 0.]
s: [0. 0. 0.08333333 0. 0. 0.08333333]
f: [0. 0.08333333 0. 0. 0. 0.08333333]
a: [0.08333333 0. 0. 0. 0.08333333 0.]
d: [0. 0. 0.08333333 0.08333333 0. 0.]]

Figure 4.1.2: the confusion matrix for one run.

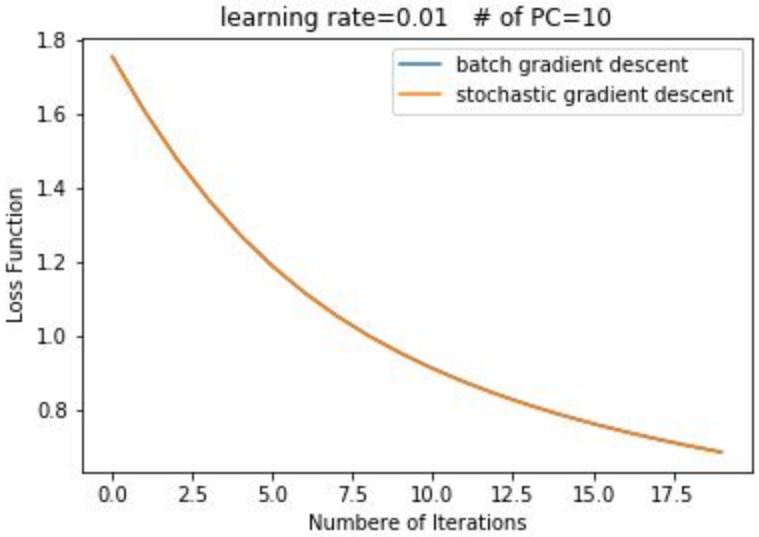

Figure 4.2.1: The figure comparing batch gradient descent and stochastic gradient descent vs. numbers of iterations are displayed here. Two lines are almost identical so overlap.

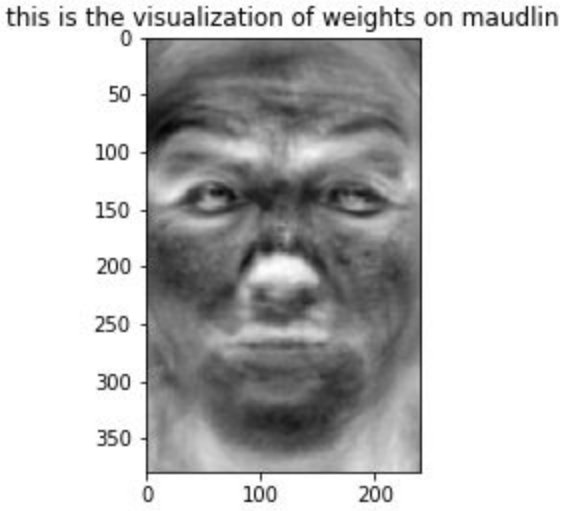

Figure 4.3.1: visualization of weights on maudlin, saved as “visual_of_m.png”

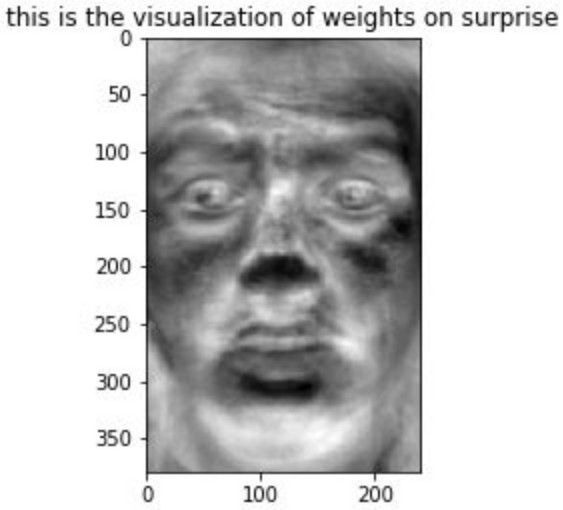

Figure 4.3.2: visualization of weights on surprise, saved as “visual_of_s.png”

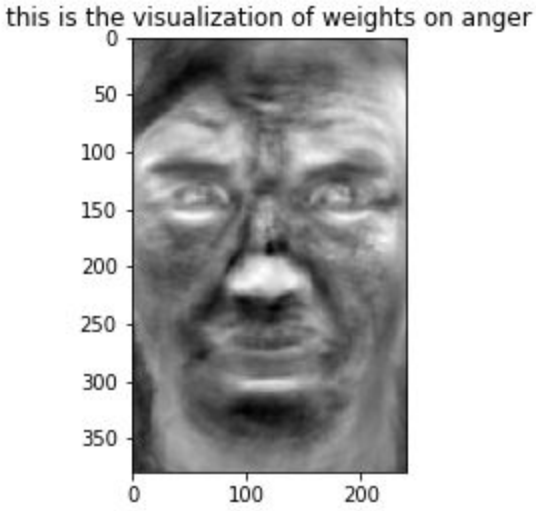

Figure 4.3.3: visualization of weights on anger, saved as "visual_of_a.png"

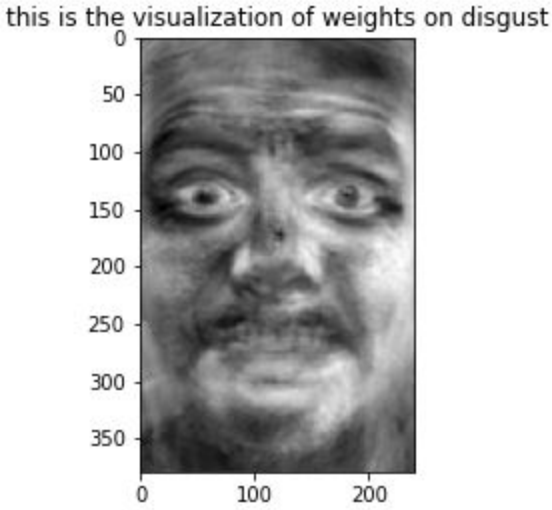

Figure 4.3.4: visualization of weights on disgust, saved as "visual_of_d.png"

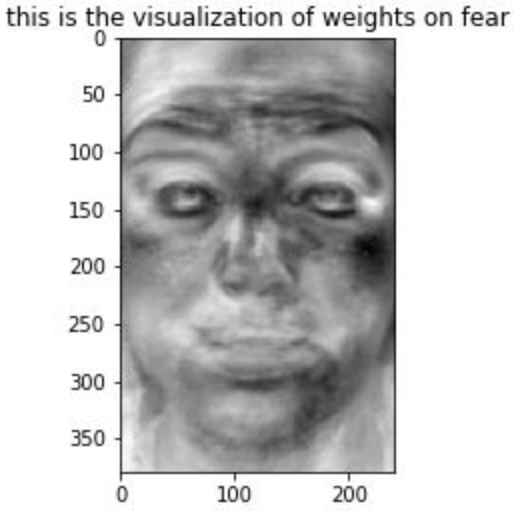

Figure 4.3.5: visualization of weights on fear, saved as “visual_of_f.png”

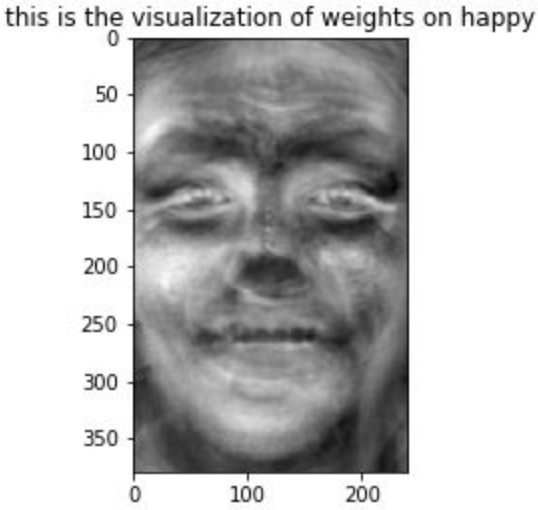

Figure 4.3.6: visualization of weights on happy, saved as “visual_of_h.png”