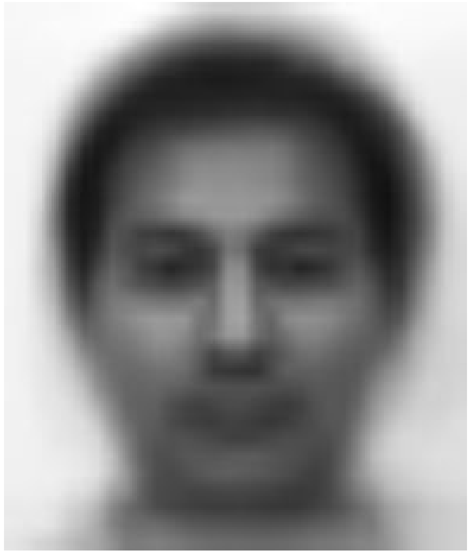
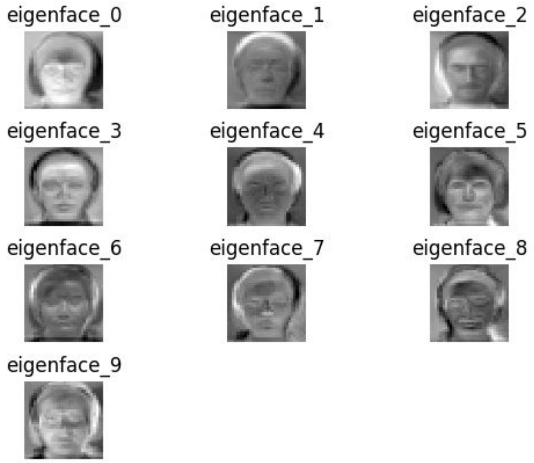
Problem Set 6: Image Classification (Faces)

Nathan Dymora: NDymora@GATech.edu

1. PCA


ps6-1-a-1.png

1. PCA



ps6-1-b-1.png

1. PCA

After performing and analyzing the facial recognition predictions, I have verified that it does perform better than just randomly selecting a label (6% chance of choosing correctly). I got mine to average out at about 16% accuracy, which is still a far way off from being ideal. I tried a variety of k values, with low values like 1 or 2 causing wild swings in accuracy (from 3% to 18%). Large k values like 1000, seemed to steady the accuracy at about 14%. I got my consistent best results at around k=10. Trying different test amounts, I strangely found that the more training you had, the worse the results. It didn’t make much intuitive sense to me, but the results seemed to verify.

2. Boosting

Iteration 1:

Train Random:	25.00%
Train Weak:	100.00%
Train Boosting:	100.00%
Test Random:	50.00%
Test Weak:	50.00%
Test Boosting:	50.00%

2. Boosting

Iteration 2:

Train Random:	50.00%
Train Weak:	100.00%
Train Boosting:	100.00%
Test Random:	25.00%
Test Weak:	50.00%
Test Boosting:	50.00%

2. Boosting

Iteration 3:

Train Random:	50.00%
Train Weak:	100.00%
Train Boosting:	100.00%
Test Random:	75.00%
Test Weak:	75.00%
Test Boosting:	75.00%

2. Boosting

Iteration 4:

Train Random:	75.00%
Train Weak:	100.00%
Train Boosting:	100.00%
Test Random:	25.00%
Test Weak:	25.00%
Test Boosting:	25.00%

2. Boosting

Iteration 5:

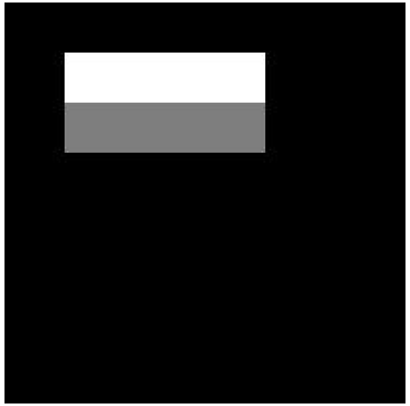
Train Random:	25.00%
Train Weak:	100.00%
Train Boosting:	100.00%
Test Random:	50.00%
Test Weak:	50.00%
Test Boosting:	50.00%

2. Boosting

All analysis was done with a 50/50 split of training/test data, and using 10 iterations through boost.

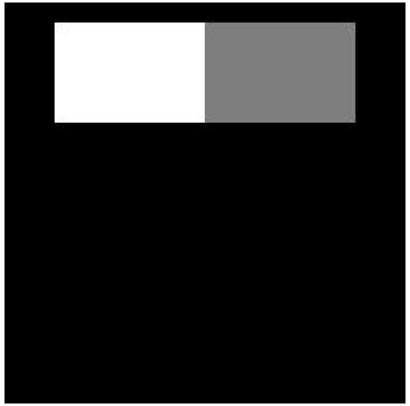
Analyzing my results, my training data came back as expected, with the random jumping around, and the weak classifier and boost staying at 100%. When it came to the actual test data however, the results seemed very similar, with only occasionally better results from the weak classifier and boost compared to picking a label randomly. When I tried higher and lower iterations, I didn’t notice much of a difference in accuracy. The surprise came when I changed the training/test split to 90/10. This kept resulting in 0% accuracy for boost, which I assumed was due to overfitting.

3. Haar-like features



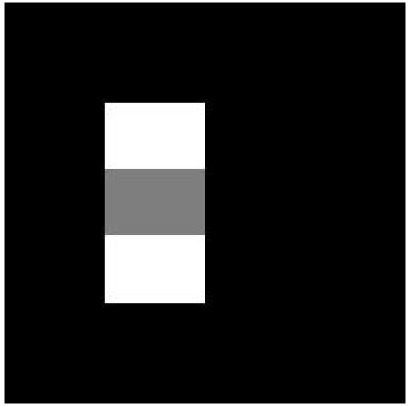
ps6-3-a-1.png

3. Haar-like features



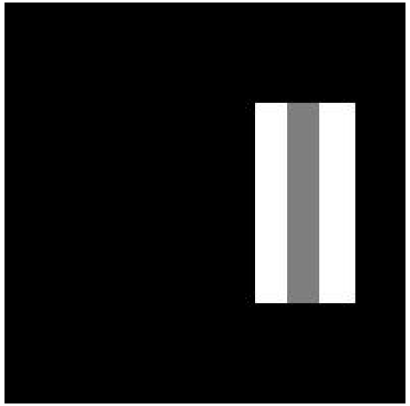
ps6-3-a-2.png

3. Haar-like features



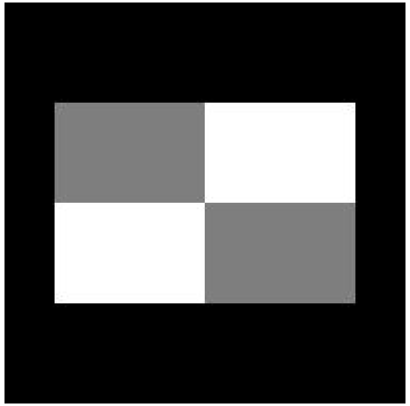
ps6-3-a-3.png

3. Haar-like features



ps6-3-a-4.png

3. Haar-like features



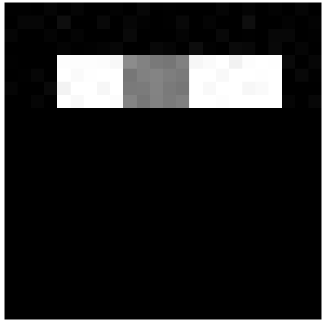
ps6-3-a-5.png

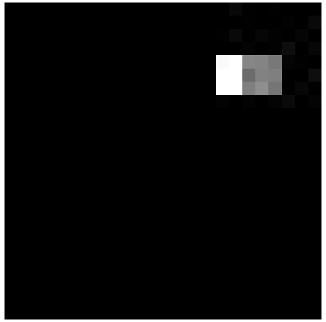
3. Haar-like features

Working with the integral images, I noticed that it helped a lot with computation time compared to the numpy sum function. Rather than looping through large quantities of numbers (which python is already a bit slow at), you only have to keep track of 4 points in order to get the sum of any points on an image.

4. Viola-Jones

Part a and b


ps6-4-b-1.png


ps6-4-b-2.png

Training accuracy:	82.86%
Prediction accuracy:	97.14%

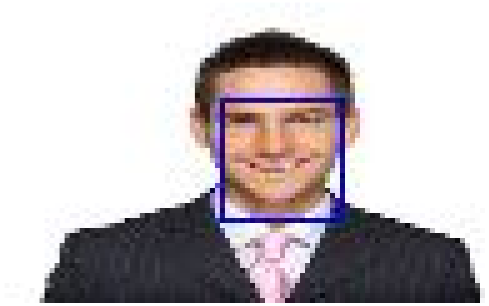
4. Viola-Jones

Part a and b

The Haar features selected by the Viola Jones trainer represent good features to use for determining faces. For example, the haar feature selected in image ps6-4-b-1 is useful since it appears to match up with how eyes are situated on the face, with a darker bar between them (like how our white eyes have a dark patch of skin between them). The second image was harder for me to determine the usefulness, but after looking at some of the positive pictures, it probably has to do with the side profile view of the face, and how the light eyes are in the front, while our darker skin falls behind them.

4. Viola-Jones

Part c



ps4-4-c-1.png