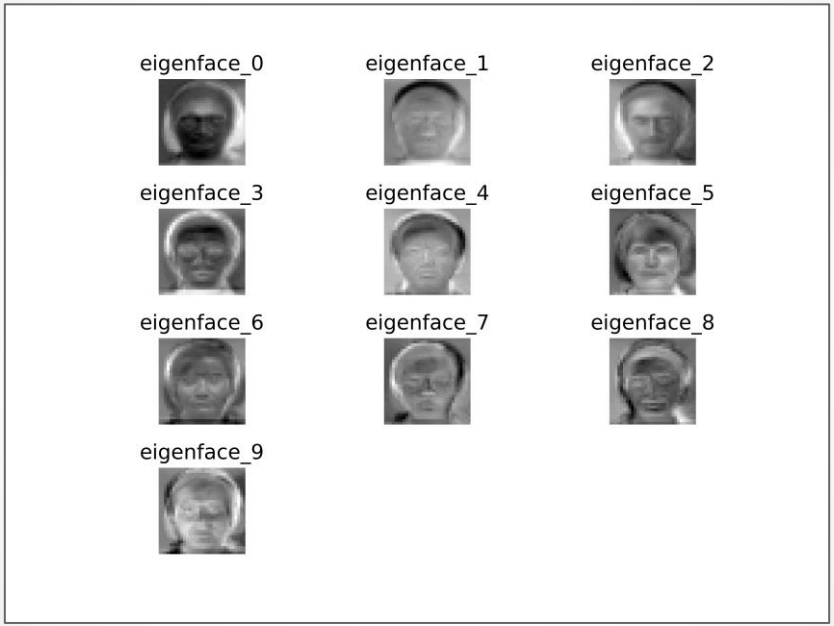
Computer Vision (2021) Problem Set #6

Josh Adams
Jadams334@gatech.edu

1a: Average face

ps6-1-a-1

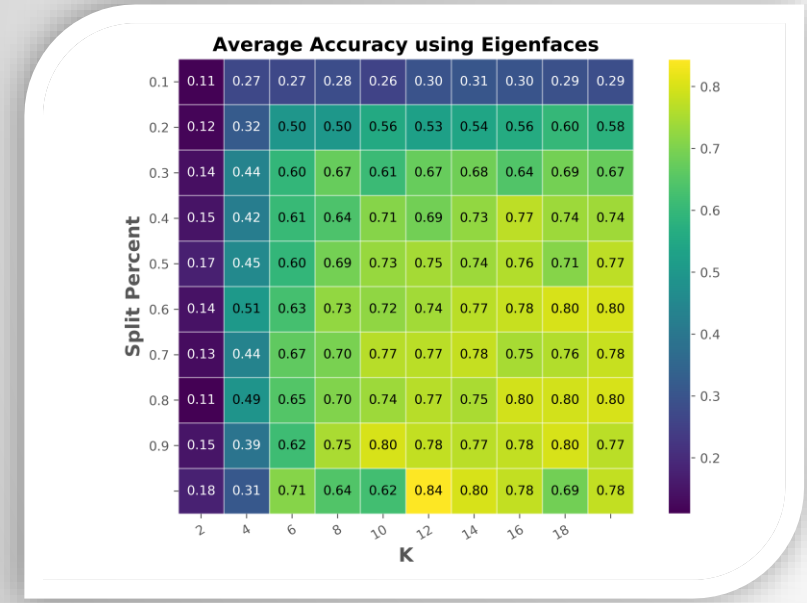
1b: Eigenvectors



ps6-1-b-1

1c: Analysis

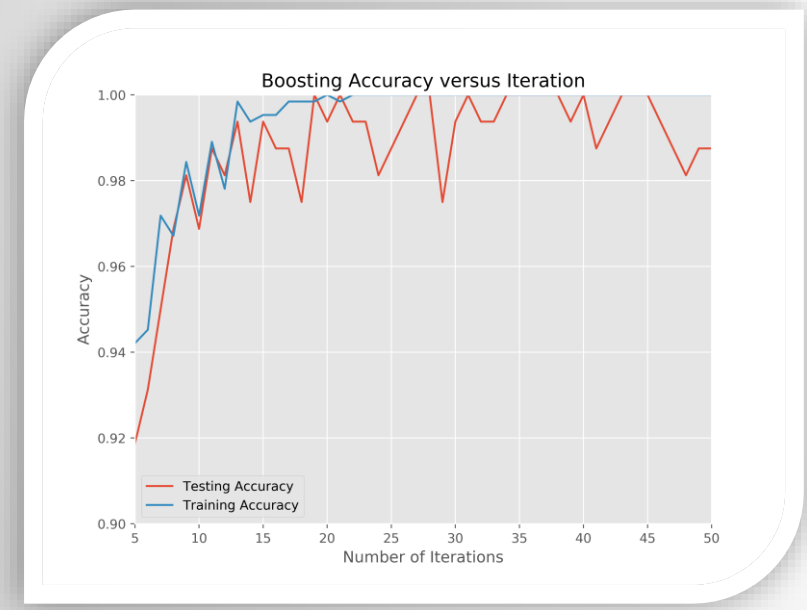
Analyze the accuracy results over multiple iterations. Do these “predictions” perform better than randomly selecting a label between 1 and 15? Are there any changes in accuracy if you try low values of k? How about high values? Does this algorithm improve changing the split percentage p?



As I vary the values of K as well as the split percentage there seems to be a positive relationship with increasing the split percentage as well as the K value. For example, looking at a split percentage of 0.1 it starts off with a 0.11% accuracy with a K of 2 and quickly increased to 0.27 with a K of 4. It appears that the highest accuracy was with a training set using 0.99 percent of the training data and a K of 12. Practically all the split percentages and various values of K, produce better accuracy than selecting a label randomly. Higher values of K tend to produce higher accuracies as well as higher split values tend to produce higher accuracies. To produce the best accuracies, increasing both the K and the split percentage P would be needed.

2a: Average accuracy

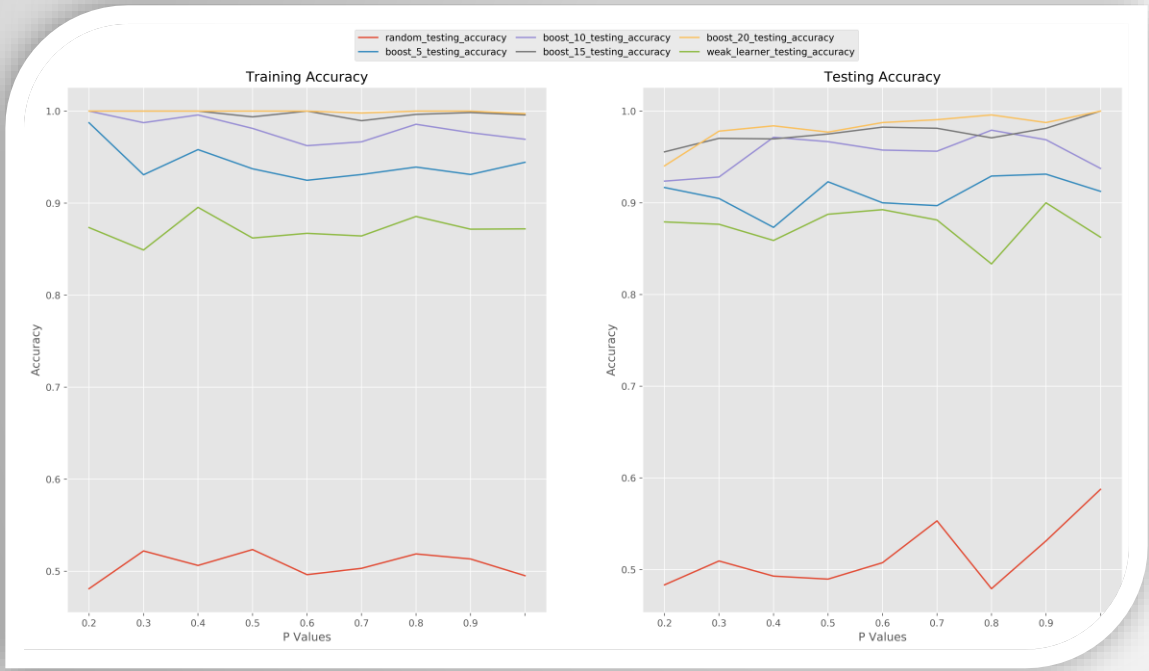
Report the average accuracy over 5 iterations. In each iteration, load and split the dataset, instantiate a Boosting object and obtain its accuracy.



The average accuracy over 5 iterations was around 90%. I wanted to check more iterations, so I ran boosting with iterations ranging from 5 to 50. Moving from 5 to 20 iterations produce the greatest increases in accuracy. Diminishing returns heavily set in after iteration number 20 as such there would be no reason to increase the number of iterations beyond that. I believe the reason is that the dataset is not terrible complicated, and the classifier is able to fit the data relatively quickly.

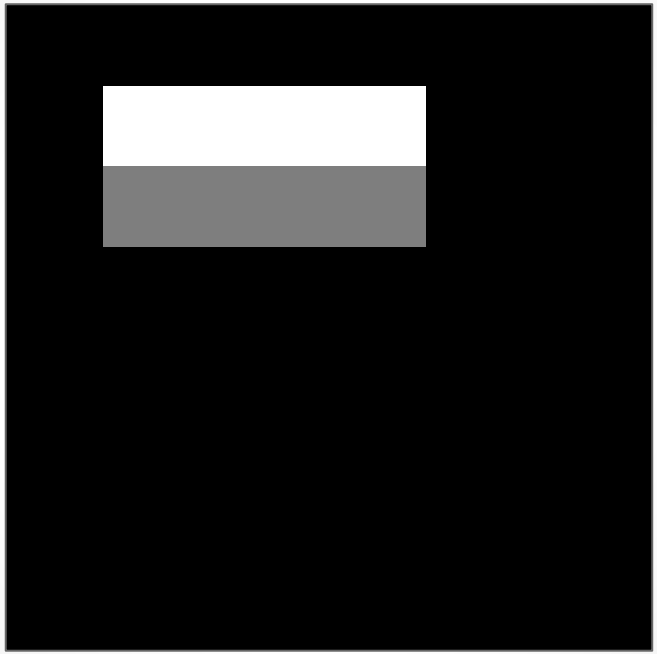
2a: Analysis

Analyze your results. How do the Random, Weak Classifier, and Boosting perform? Is there any improvement when using Boosting? How do your results change when selecting different values for num_iterations? Does it matter the percentage of data you select for training and testing (explain your answers showing how each accuracy changes)



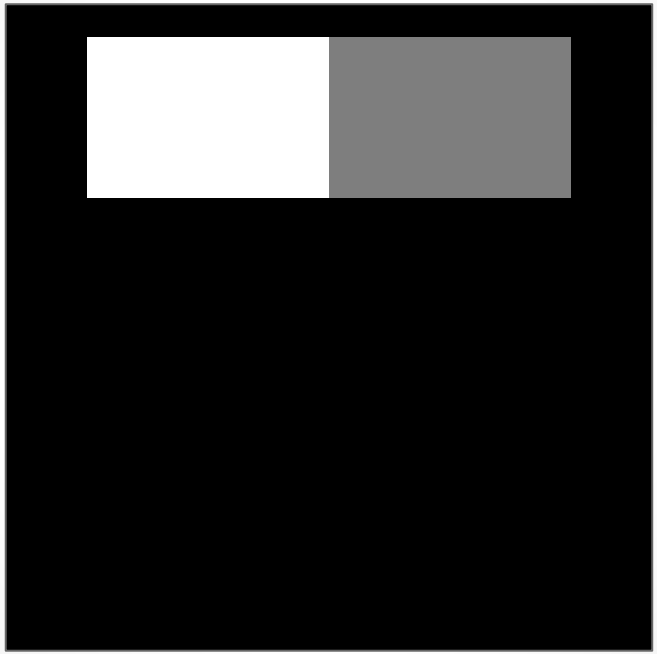
Looking at the various results from the random, weak and boosting classifiers with various P values set. The random classifier stays near 50% accuracy throughout but manages to get near 60% towards the end of the testing. I expect this was due to not averaging enough samples. Boosting is consistently performing better than the weak learner in both training and testing accuracy as each of the P values. The boosting classifiers seem to benefit more consistently from the number of iterations versus the amount of training data it receives. boost_10_testing_accuracy means a boosting classifier with 10 iterations on the testing data.

3a: Haar Features



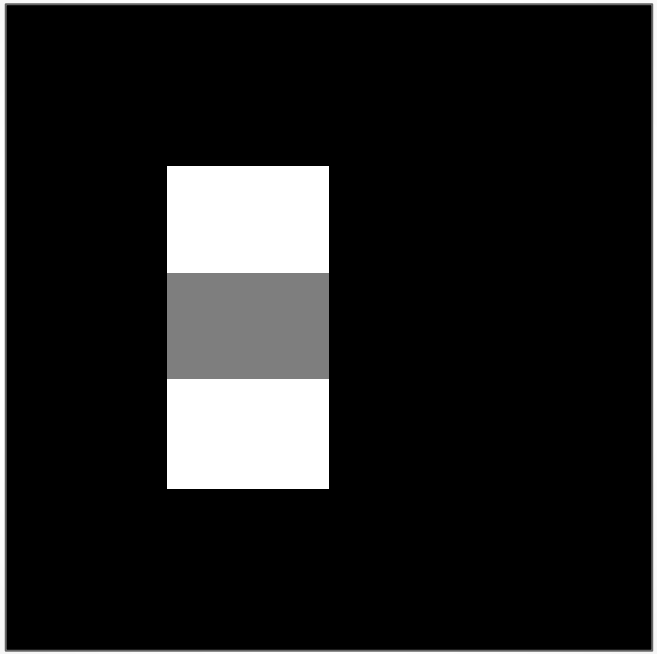
ps6-3-a-1

3a: Haar Features



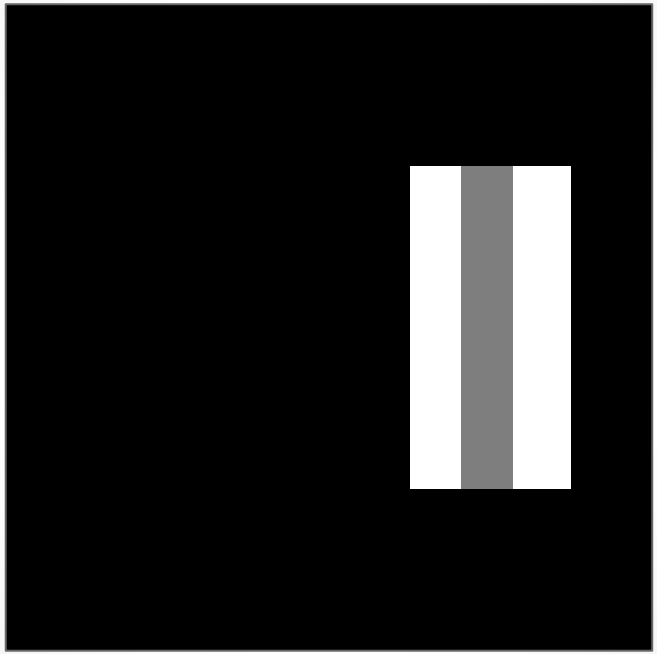
ps6-3-a-2

3a: Haar Features



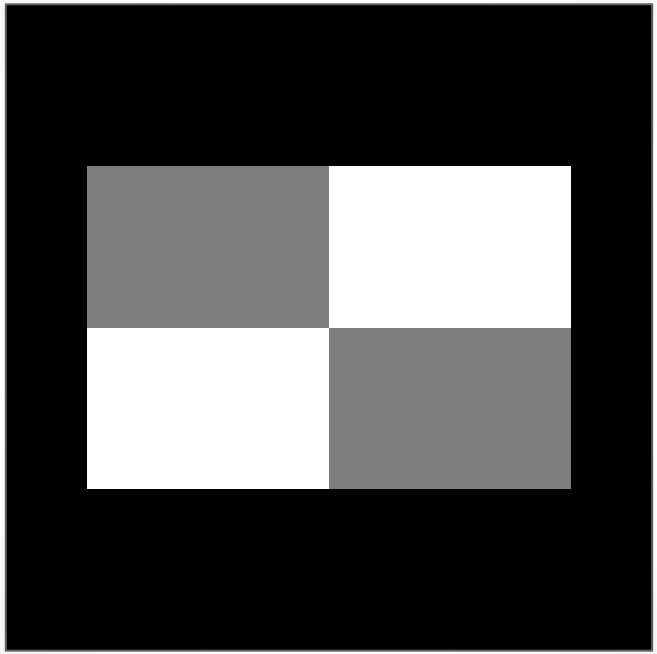
ps6-3-a-3

3a: Haar Features



ps6-3-a-4

3a: Haar Features



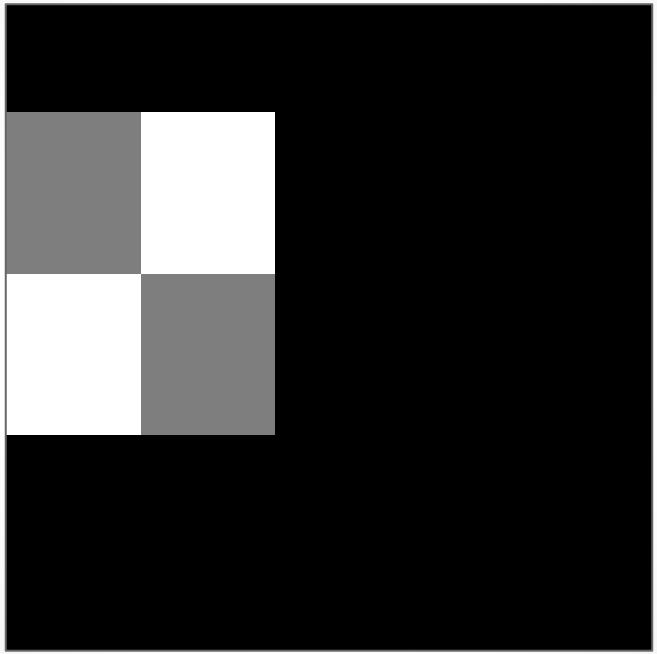
ps6-3-a-5

3c: Analysis

How does working with integral images help with computation time? Give some examples comparing this method and np.sum.

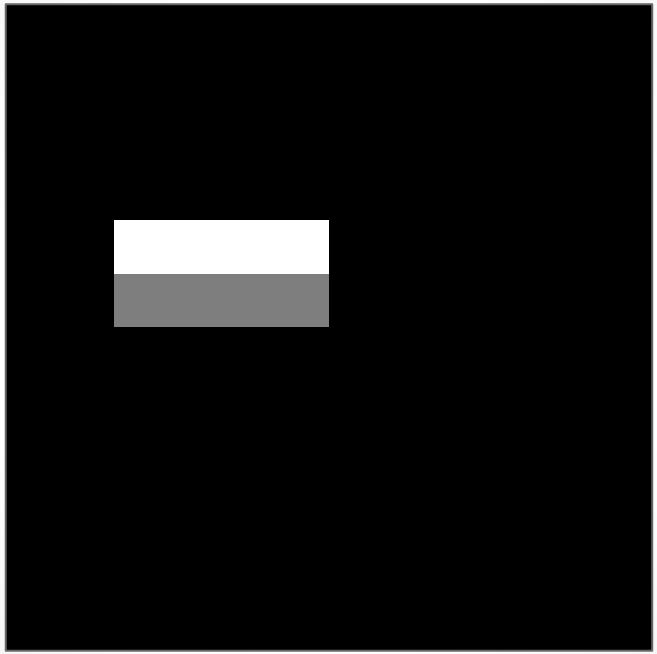
Working with the integral images dramatically improves computation time in this problem. The reason is that for every gray or white segment in feature, we would have to run np.sum. This would increase computation time considerably as there are thousands of features and each feature can have multiple gray or white segments. You just need to know four points of a segment and this may take minor operations such as addition, subtraction. Then using the integral image, can directly access the summation of those areas in constant time, O(1). An example of this is using the tests for ps6 and Haar features, using np.sum resulted on average taking 447ns to complete while the integral image took on average 276ns. This improvement is just for one feature and already nearly reduces the computation time in half. This improvement is made more significant as the number of features used is increased.

4b: Viola Jones Features



ps6-4-b-1

4b: Viola Jones Features



ps6-4-b-2

4b: Analysis

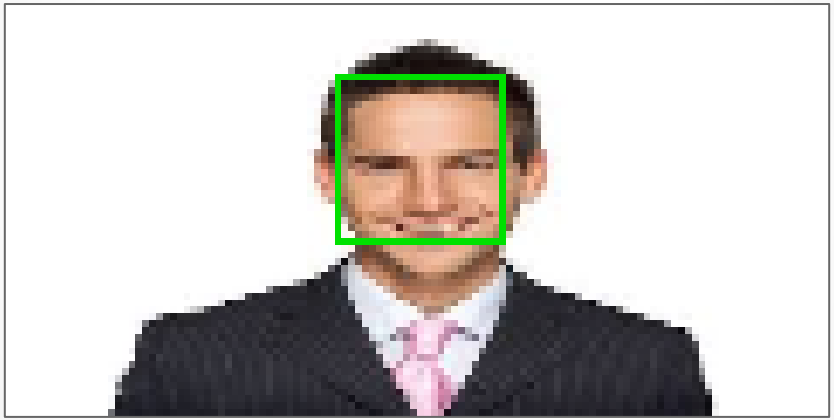
Report the classifier accuracy both the training and test sets with a number of classifiers set to 5. What do the selected Haar features mean? How do they contribute in identifying faces in an image?

Setting the number of classifiers to 5, the training accuracy was 98% and the testing accuracy was 80%.

```
Connected to pydev debugger (build 203.7717.81)
 -- compute all scores --
 -- select classifiers --
 -- training complete --
Prediction accuracy on training: 0.98%
Prediction accuracy on testing: 0.80%
```

The features selected were the best features to distinguish between images which contain a face and those that do not. The feature ps6_4_b_1 does not provide to me much intuition behind its generation or value. The feature ps6_4_b_2 on the other hand does. I can see how when this feature is aligned in a specific fashion, faces would be distinguishable. An example of this would be the eyes and the nose areas of the face. These two areas typically have very different aspects. The feature can capture those different aspects well enough due to the feature being split horizontally. The feature ps6_4_b_1 does accomplish something similar but the extra white and black squares on the left side are not very intuitive to me. In essence I would expect these two feature to accomplish very similar tasks.

4c: Viola Jones Face Recognition



ps6-4-c-1