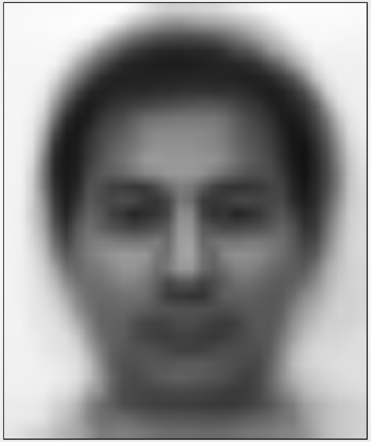
Computer Vision
Spring 2020
Problem Set #6

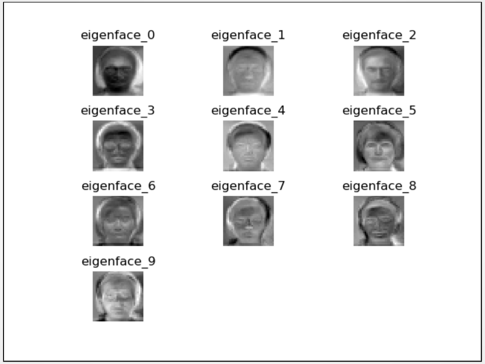
Andrew Parmar
aparmar32@gatech.edu

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?

A random selection should have a mean accuracy of 50%. The predictions with the naive classifier definitely perform better. The average performance over six runs using a split p=0.8 and k=7 is 64%. Using k values lower than 5 showed a reduction in accuracy. The accuracy plateaus around k=7.
The p value doesn’t appear to be as sensitive to the accuracy. Using a value of p=0.4 still provided an average accuracy of 64%, and accuracy values of 50% for p as low as 0.2. Raising p to greater than 0.5 showed no significant improvements in accuracy.

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.

% Accuracy (Average of 5 runs) - Random and Weak

random	weak
51.0	87.0

% Accuracy (Average of 5 runs) - Boosting

boosting-accuracy	boosting-num_iter
87.125	1
86.5	2
88.375	3
89.5	4
92.625	5
93.25	6
95.375	7
94.75	8
96.75	9
96.0	10
99.0	15

% Accuracy (Average of 5 runs) - Changing p fixed num_iterations

p-split	random	weak	boosting
0.1	49.03	85.39	87.89
0.2	51.47	86.22	91.41
0.3	49.50	85.46	89.32
0.4	50.75	87.04	91.62
0.5	48.40	86.40	91.35
0.6	49.00	88.25	92.19
0.7	49.58	85.75	91.67
0.8	47.38	85.88	92.62
0.9	52.75	88.25	92.75

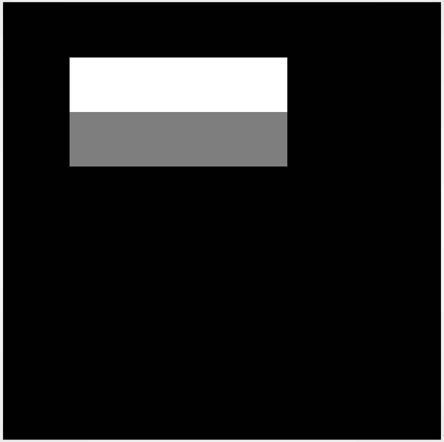
2a: Analysis

Analyze your results. How do the Random, Weak Classifier, and Boosting perform? Is there any improvement when using Boosting?

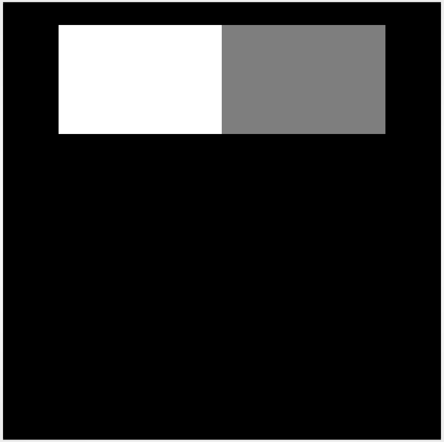
How do your results change when selecting different values for num_iterations? Does it matter the percentage of data you select for training and testing (explain your answers showing how each accuracy changes).

As expected, the random classifier has an average accuracy close to 50%. The weak classifier does a significantly better job of around 87%.
Boosting as we expect, performs the best. The number of weak classifiers used to create the boosting classifier (num_iter) is directly proportional to the boosting classifiers accuracy. The default value of 5 iteration provides a 5% increase over a single weak classifier. While using 10 iterations yields 8% better performance. As expected, the higher iteration do take longer to run, so there is a trade-off between accuracy and speed.
As the p-values don't play a part in the random classifier, we don't see any effect of changing the data split value here. The weak classifier also seems relatively consistent in accuracy - any variation is small that it is not discernible from noise. The boosting classifier however, shows a minor but noticeable improvement with increasing training data.

3a: Haar Features

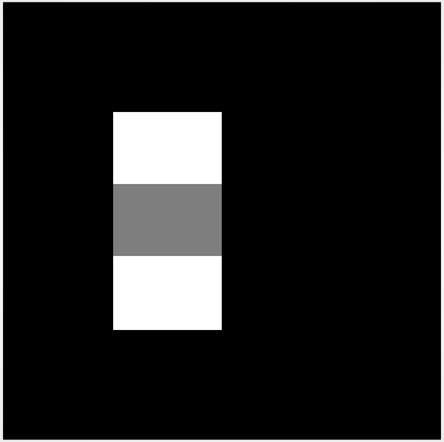

ps6-3-a-1

3a: Haar Features



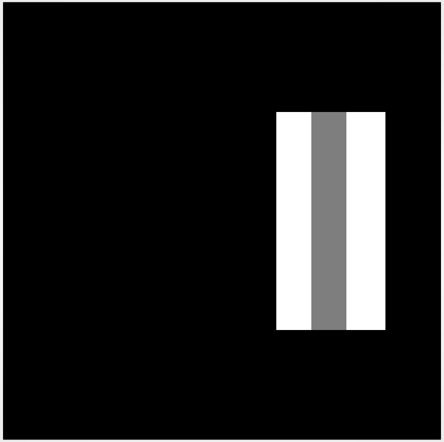
ps6-3-a-2

3a: Haar Features

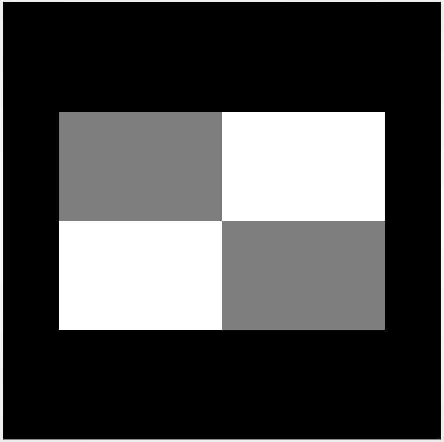


ps6-3-a-3

3a: Haar Features


ps6-3-a-4

3a: Haar Features



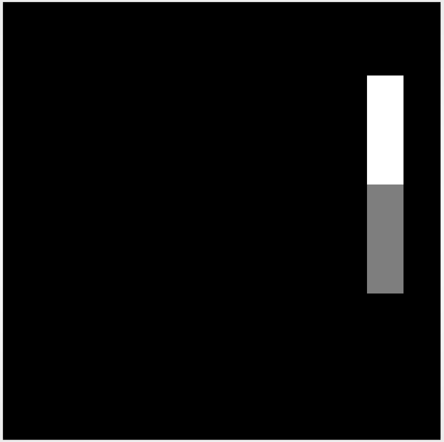
ps6-3-a-5

3c: Analysis

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

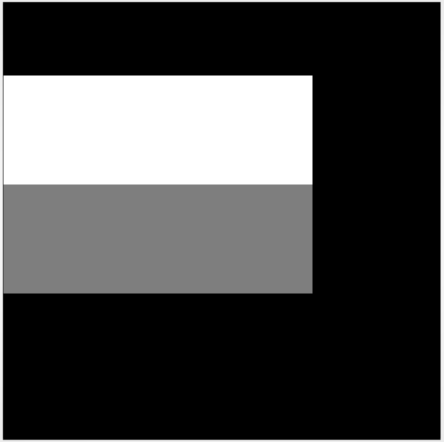
Integral images allow us to keep computation of areas in constant time regardless of the size of the image. The np.sum() computation has to compute sums over the complete sub-matrix while the integral image method has to just retrieve the same number of elements regardless of the matrix size. There is an upfront cost of computing the integral image which will be proportional to image size. However this is minimal in a situation where thousands of summation calculations need to be performed. These examples show the difference in computation times using the two methods and matrices of different sizes.
HaarFeature type (2,2)
Matrix size = (10000, 10000) Summation size = (8000, 8000) np.sum: 0.0636899471, integral_image: 0.0000143051
Matrix size = (20000, 20000) Summation size = (18000, 18000) np.sum: 0.7915508747, integral_image: 0.0102751255
Matrix size = (40000, 40000) Summation size = (28000, 28000) np.sum: 15.1851959229, integral_image: 0.0048100948

4b: Viola Jones Features



ps6-4-b-1

4b: Viola Jones Features



ps6-4-b-2

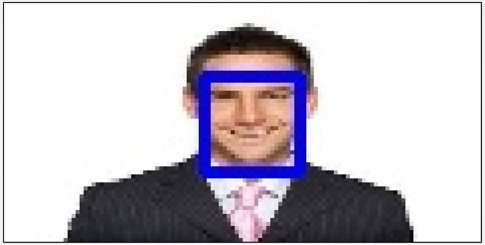
4b: Analysis

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?

Prediction accuracy on training: 97.14%
Prediction accuracy on testing: 74.29%
The selected features represent the most common brightness patterns in the faces within the training set.

They are essentially represented in the form of gradient filters that can be applied to an image to find area that satisfy the brightness pattern. These match areas within the image are the most probable areas for faces.

4c: Viola Jones Face Recognition


ps6-4-c-1