Project 2 Human Face Detection using Boosting

Xiaohan Wang

1. Construct weak classifier

In this part, we use optimized method to select weak classifier with lowest weighted error:

1) Sort activations with increasing order, and also sort data's weights and labels correspondingly.
2) Create two prefix sum arrays: sum_of_pos_weight and sum_of_neg_weight. For first array, at specific index, the value represents the total sum of weights of all elements before this index (including it) whose labels are 1. The second array has similar meaning.
3) Traverse all values of activations from left to right as current threshold. Calculate error and update polarity and threshold of the weak classifier. Here, we need to assume polarity as 1 and -1 respectively:
   - If polarity is 1, then all elements before current index are predicted as -1, after it as 1. Then the error would be:

$$err = \sum_{j=0}^{i-1} weight_j + \sum_{k=i}^{n-1} weight_k \,, where\ label_j = 1, label_k = -1$$

$$= sum\_of\_pos\_weight[i-1] + (sum\_of\_neg\_weight[len-1] - sum\_of\_neg\_weight[i-1])$$

   - If polarity is -1, then it would be opposite of the above situation.
4) For each possible threshold, we need to compare two kinds of errors with the global min_error. Then update the min_error and the polarity and threshold correspondingly.

2. Implement AdaBoost

a) Haar filters

After selecting all weak classifiers for the strong classifier, we sort those weak classifiers according to their voting weights α_t in decreasing order. Then we can display top 20 haar filters as below. Those 20 filters represent most apparent features of the faces in the training data.

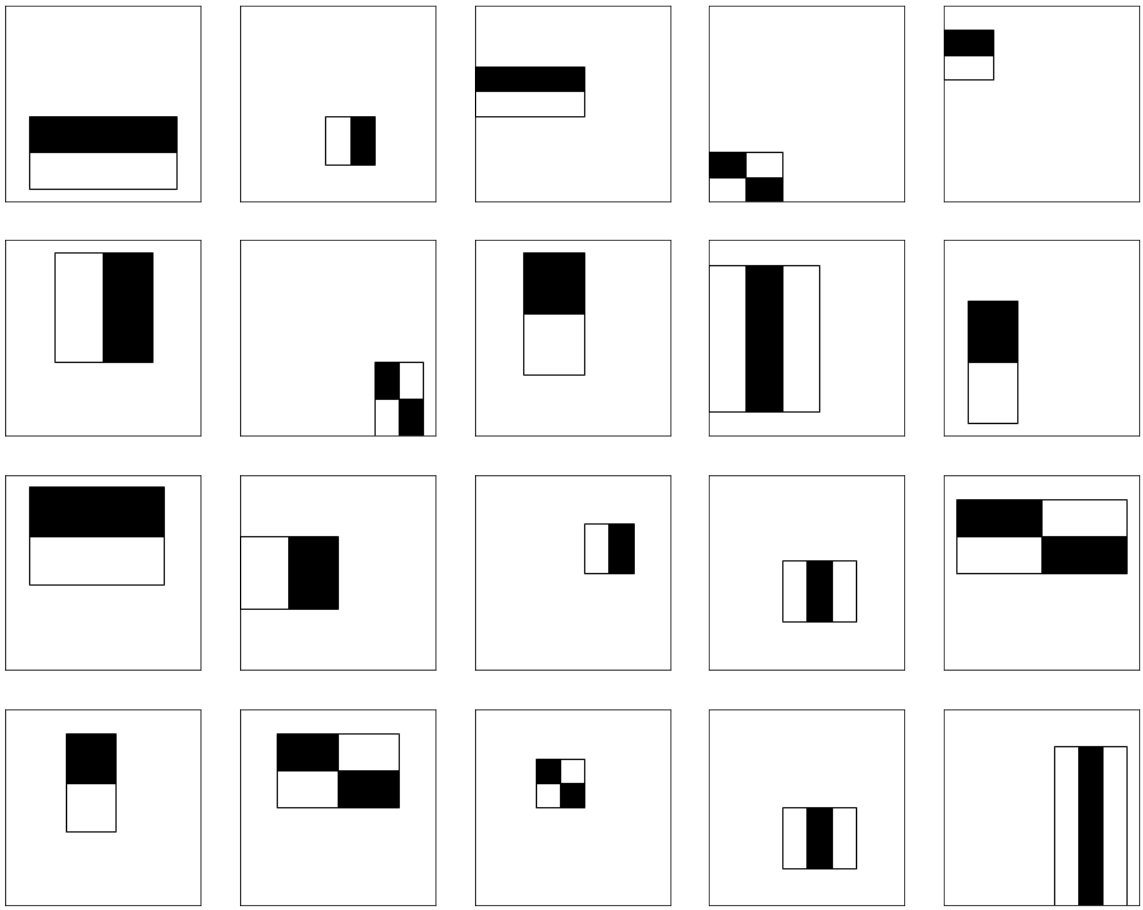

Figure 1. top 20 haar filters after boosting

The corresponding voting weights α_t are as below:

Table 1. voting weights of top 20 haar filters

1	0.4770	6	0.2418	11	0.2082	16	0.1898
2	0.3812	7	0.2301	12	0.2027	17	0.1891
3	0.3153	8	0.2278	13	0.1970	18	0.1863
4	0.2552	9	0.2229	14	0.1914	19	0.1858
5	0.2551	10	0.2205	15	0.1906	20	0.1845

In the above table, the voting weights are listed decreasingly. From the equation of voting weights $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}$, we can find that the bigger the α_t is, the smaller the $\epsilon_t\ (h_t)$ is, and the more significant this haar filter is.

b) Training error of strong classifier

In this part, we can pass each training data to each weak classifier of the strong classifier. Then, we can get a list of predictions (1, number of training data) by multiplying each weak classifier's voting weight and prediction of each weak classifier. Next, we compare each error with 0 and set the new value as 1, if old value is greater than 0, or -1 for the opposite. Then, we can compare each prediction with corresponding true labels. If it's same as true label, then continue. Or we can add 1 to the error of strong classifier. Below is the figure of training error of strong classifier:

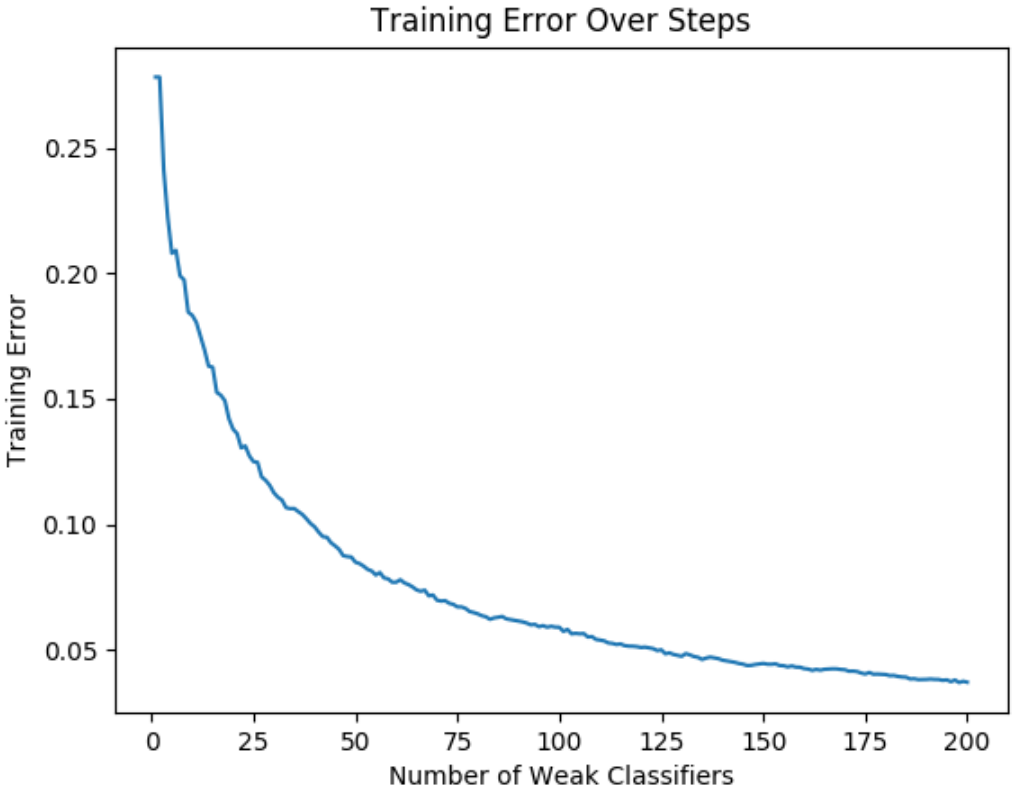

Figure 2. training error of strong classifier over steps (T = 200)

In the above figure, we find that as step increases, the training error decreases sharply. It indicates that with adding more weak classifiers and adjust weights of the data sample, the strong classifier is greatly improved, and the error rate decreases apparently. It's intuitive. For example, we have two weak classifiers, and each has error rate 40%. After adding them together with the even voting weights, the error rate becomes 40% * 40% = 16% theoretically. It greatly reduces the error.

c) Training errors of weak classifiers

In each round of 0, 10, 50, 100, we record the training errors of top 1000 weak classifiers, and then plot the curve as below:

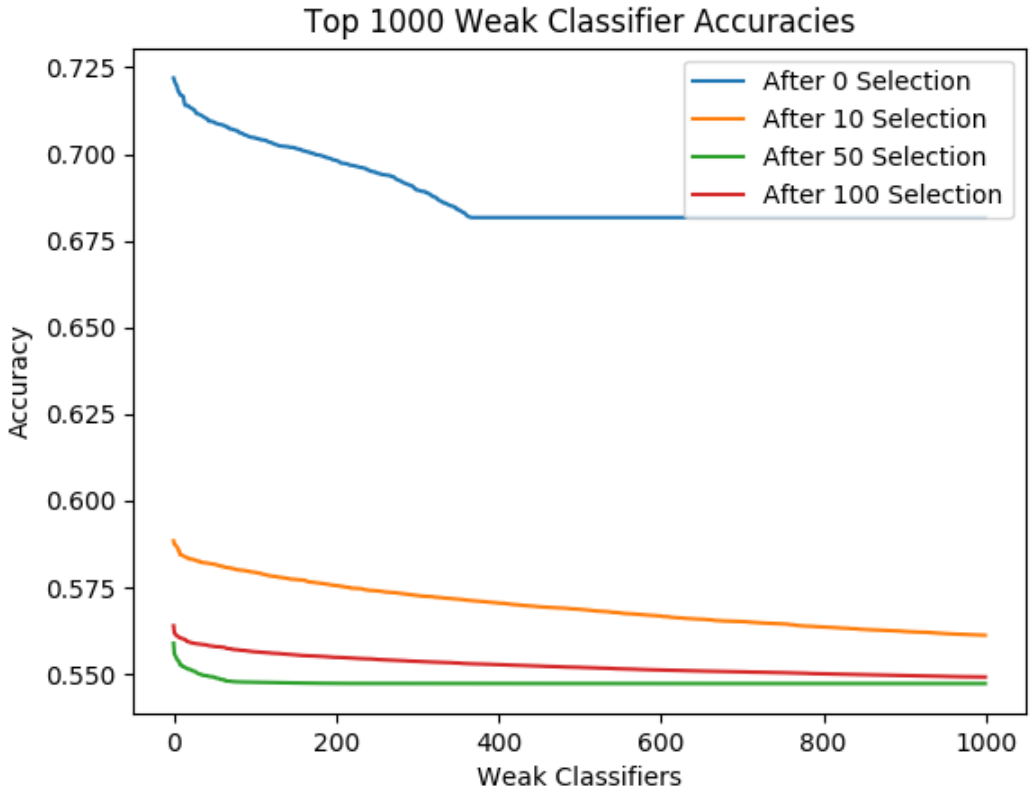

Figure 3. training errors of top 1000 weak classifiers over steps (T = 0, 10, 50, 100)

From the above figure, we find that the general trend of these four curves is decreasing. It indicates that the weak classifiers with larger No. have lower accuracies. We sort those weak classifiers first, so the ones with higher accuracies come first. Besides, we also find that as step increases, the accuracy decreases to 0.55. As more weak classifiers in, we add more errors in, because the weak classifiers with high accuracies are selected with priority. And it would also be easier to compensate the error at first with less weak classifiers since each classifier functions with more power. And later with more classifiers already in, add one classifier cannot change the situation much.

d) Histograms

In each round of 10, 50, 100, we record strong classifier scores and plot histograms as below:

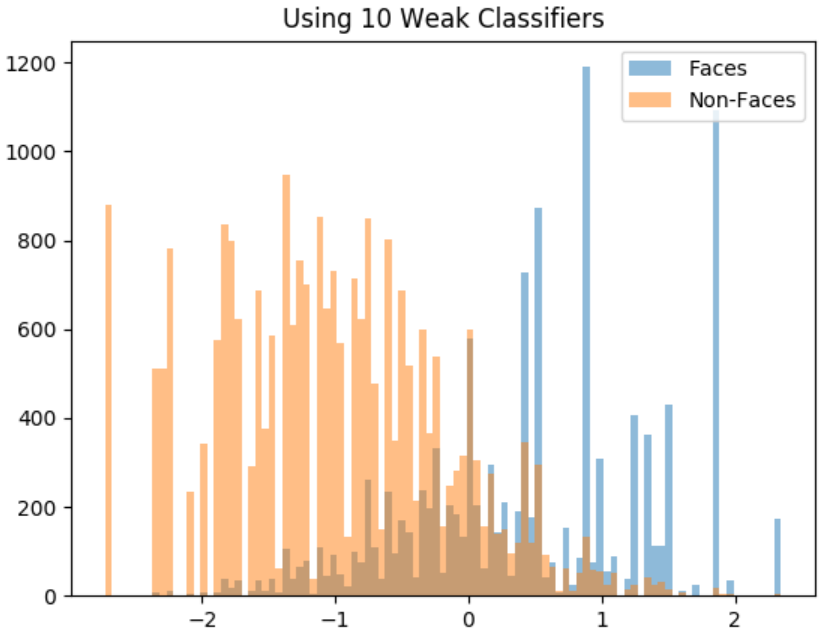

Figure 4. histograms of positive and negative populations over F(x) (T = 10)

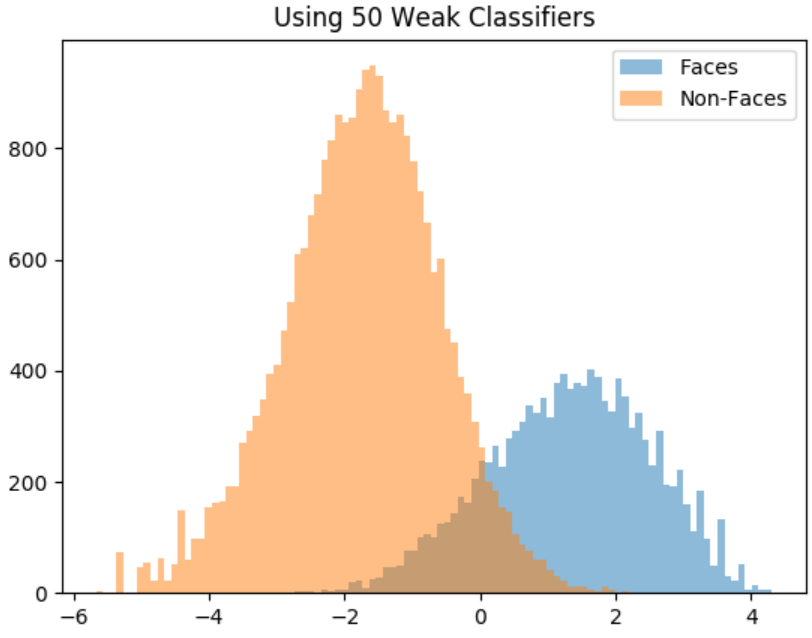

Figure 5. histograms of positive and negative populations over F(x) (T = 50)

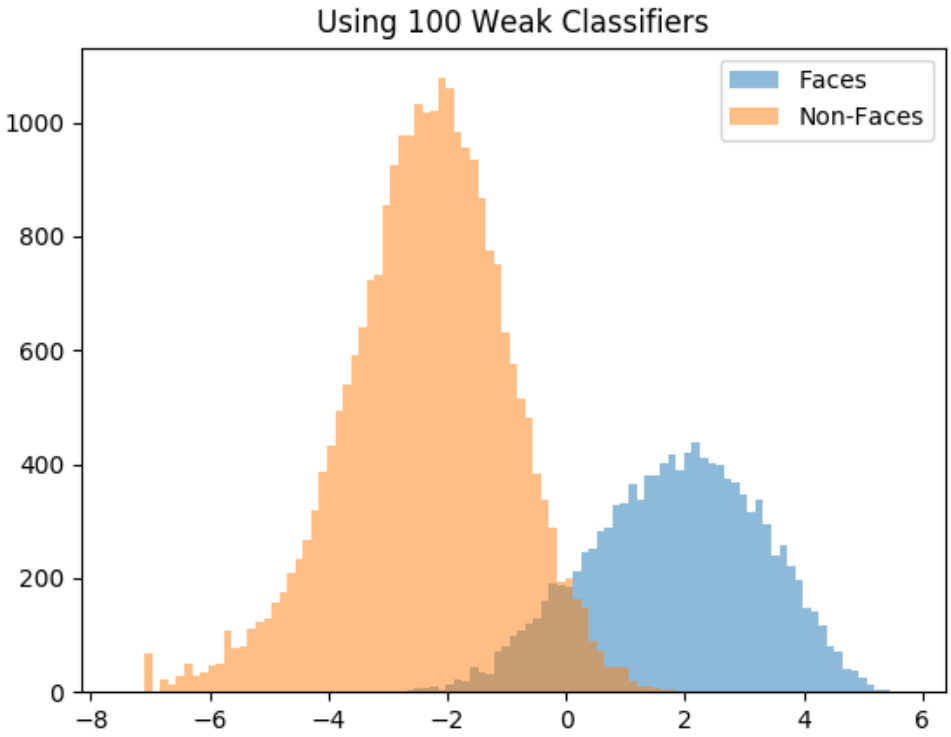

Figure 6. histograms of positive and negative populations over F(x) (T = 100)

From three figures above, we can find that as step goes, the histograms representing faces and non-faces get more apart, and for each part, the data becomes more centralized. Using 10 weak classifiers, the predictions distributed sparsely from -2 to 2. As more weak classifiers in, the shape becomes clearer with two peaks. And the altitude of the peak increases, as for 50 weak classifiers, they are about 900 and 400, and for 100 weak classifiers, they become 1000 and 420.

e) ROC

Based on histograms above, we can plot corresponding ROC curves for steps 10, 50, 100 as below:

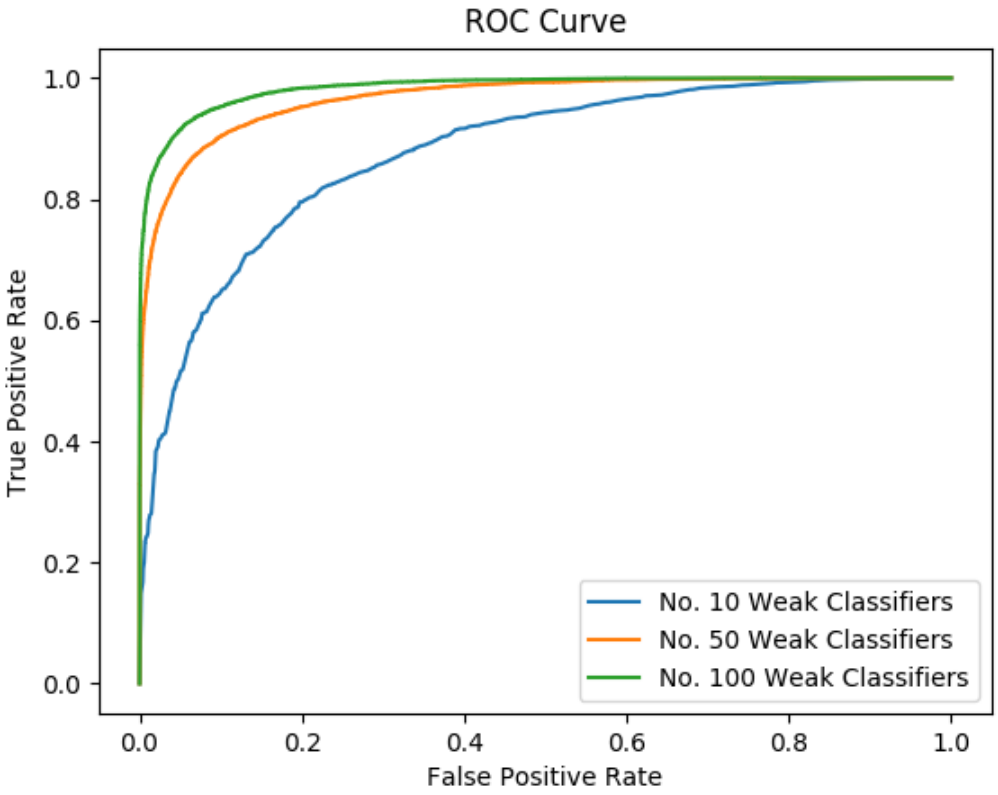

Figure 7. ROC curves for T = 10, 50, 100

In the above figure, we find that as more weak classifiers in, the results are getting better as the curves lean to true positive area with high rates. It makes sense because more weak classifiers can reduce the error continually.

f) Detections

Using the trained AdaBoost classifier, we can detect faces on testing images. Below are the detection results. The faces are almost detected, but there also exists many mis-detections.

Figure 8. detected faces on test image 1

Figure 9. detected faces on test image 2

Figure 10. detected faces on test image 3

g) Hard negative mining

In this part, we run two hard negative images without faces using the trained classifier. Then add hard negatives to negative population in the training set and re-train the model. Below are the new detection results. Compared with the figures in previous part, we can find that the mis-detections have been reduced.

Figure 11. new detected faces on test image 1

Figure 12. new detected faces on test image 2

Figure 13. new detected faces on test image 3

3. Implement RealBoost

a) Histograms

In this part, we use top T = 10, 50, 100 features in step (c) to implement the RealBoost algorithms. Then we can plot histograms of the positive and negative populations over F(x) as below.

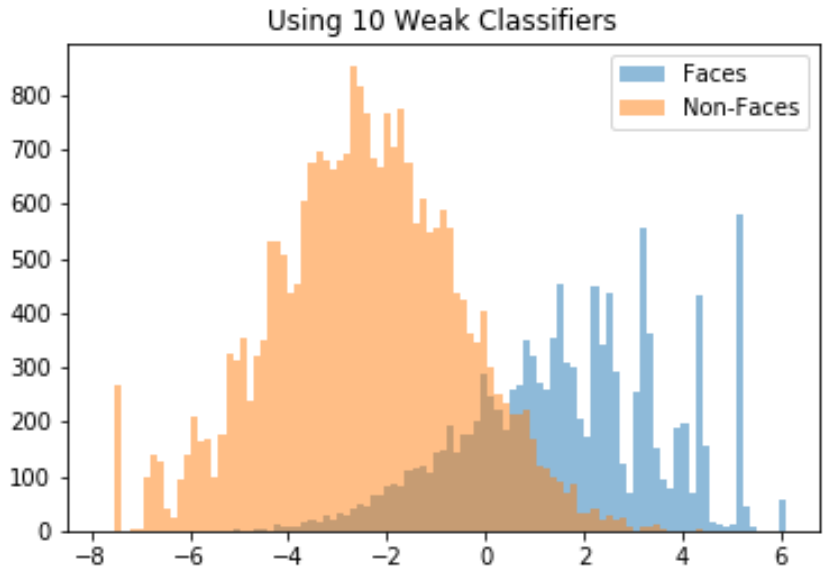

Figure 14. histograms of positive and negative populations over F(x) (T = 10, RealBoost)

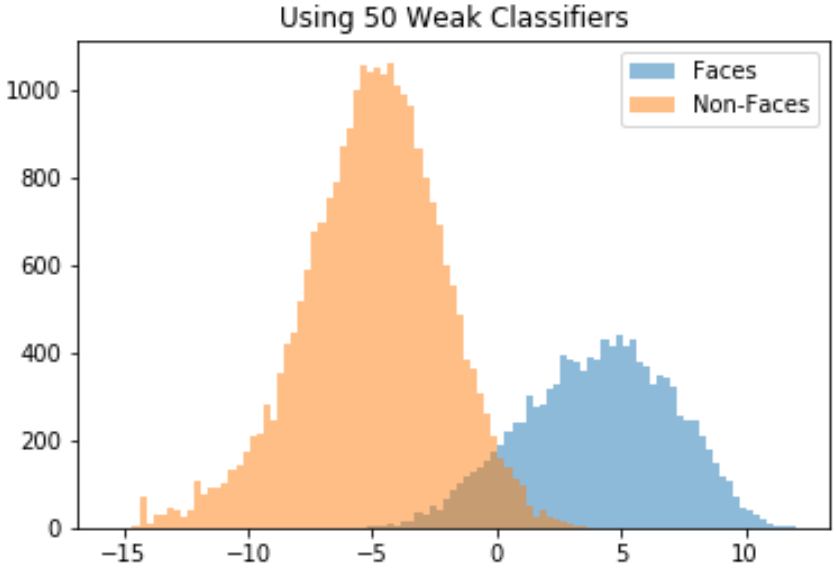

Figure 15. histograms of positive and negative populations over F(x) (T = 50, RealBoost)

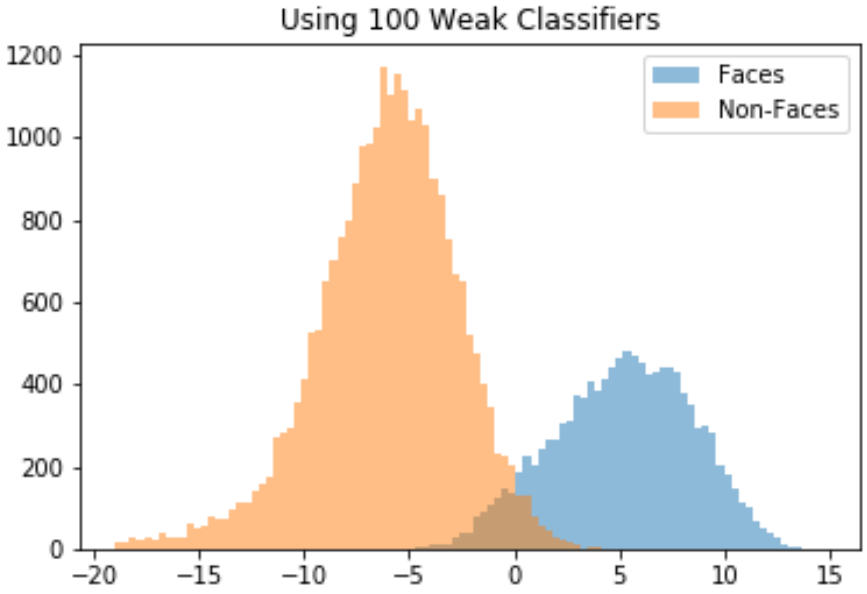

Figure 16. histograms of positive and negative populations over F(x) (T = 100, RealBoost)

In theory, RealBoost will have better results than AdaBoost since AdaBoost only fits the classifier using weights on the training data while RealBoost fits the classifier to obtain a class probability estimate. It makes RealBoost more accurate. From the figures above, it can be seen that faces and non-faces are even more apart and have even less overlap.

b) Roc

Based on histograms above, we can plot corresponding ROC curves for steps 10, 50, 100 as below. The results are similar to AdaBoost.

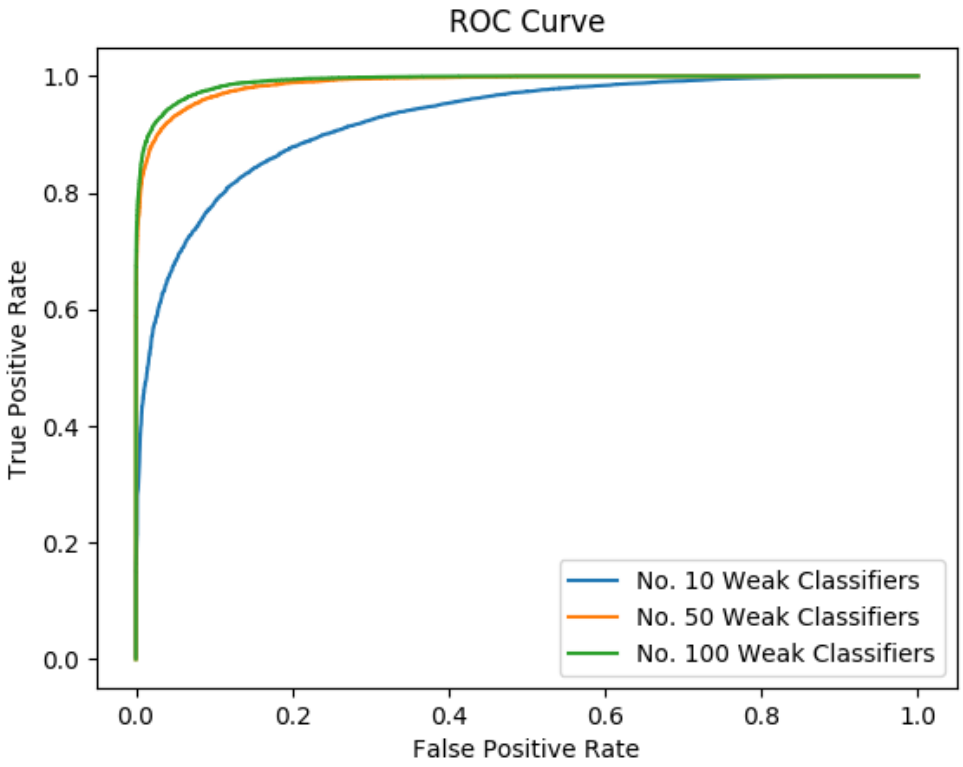

Figure 17. ROC curves for T = 10, 50, 100 (RealBoost)

The ROC curve gained by RealBoost outperforms the curve gained by AdaBoost. All these three curves are more approaching to the left upper corner than curves of AdaBoost which indicates that the results have higher accuracies.