Assignment 4: Detecting Human Eyes

Jia Ming Liang
December 9, 2014

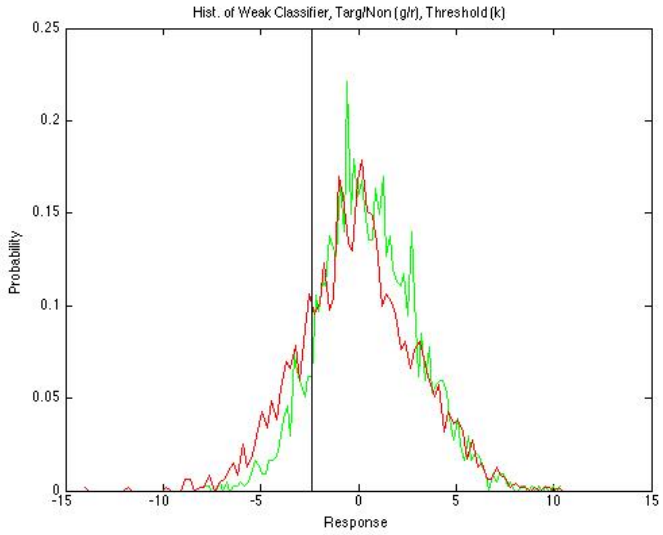
1 WEAK CLASSIFIERS

To compute θ_1 and θ_2 in trainStump.m, we minimize the error (err = $\frac{\sum_k w_k I(y_k \neq h(\vec{x}_k, \vec{\theta}))}{\sum_k w_k}$). We can see that err is smallest when $I(y_k \neq h(\vec{x}_k, \vec{\theta}))$ is minimized. That means we want our classifier to return as little false positive and false negative as possible.

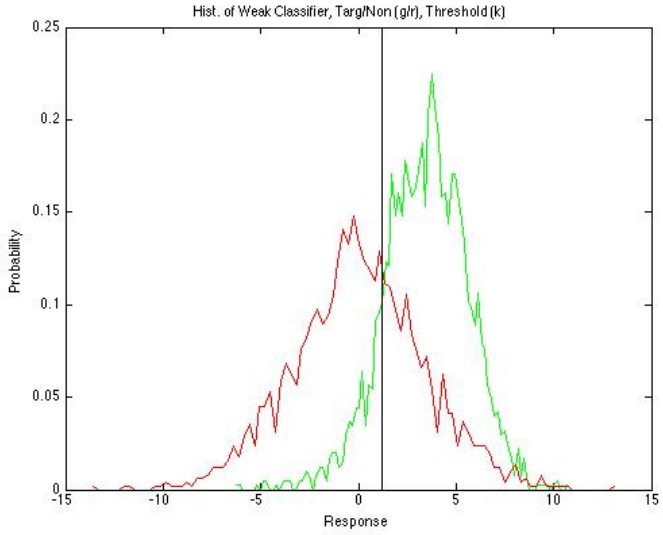
To find the correct θ_1, we first need to extract the target weight and none target weight. We then sort (use sortrows function) these weights by ascending order of u($\vec{f}^T \vec{x}$) (we use r to represent this in Matlab). After we do two cumsums in opposite order for these two sorted weights, then add the two cumsums up. Finally we determine the index for the minimum of the two cumsums sumation. θ_1 is equal to r[index] (for r in sorted ascending order).

Based on equation (1) in the handout, the cumsum order actually determined by the parity. For parity equal to 1 ($\theta_2 = 1$), we do cumsum on the none target weight from smallest to largest r (these are false negative sum). Then we do the opposite direction cumsum for target weight (false positive sum). We add these two cumsum up then find the minimum error and its index to find θ_1. For parity equal to -1 ($\theta_2 = -1$), we do cumsum on the target weight from smallest to largest r. Then we do the opposite direction cumsum for none target weight. Once we compute θ_1 for each parity, we take the minimum one to be our final θ_1.

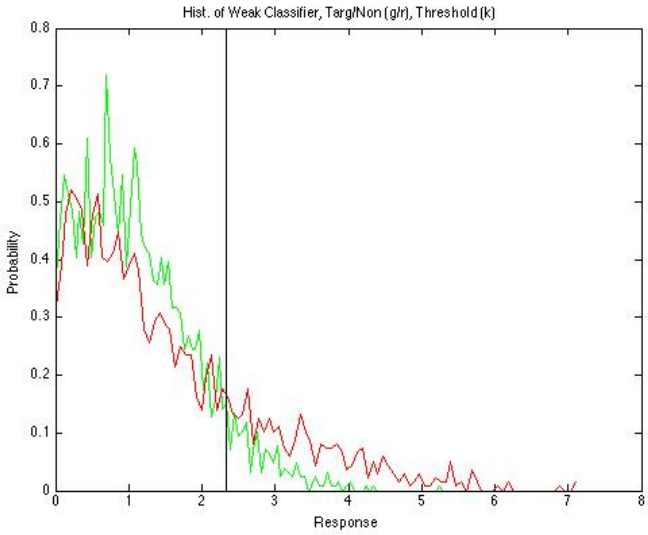
Below are results from running testSTump.m for different sets of parameters:



True/False Pos. Rates: 0.888796, 0.792600, θ_1: -2.387453, θ_2: -1, err: 0.414753
pFeat.sigma: 2, pFeat.deriv: 1, pFeat.theta: pi/4, pFeat.abs: false, x0: 5, y0: 15



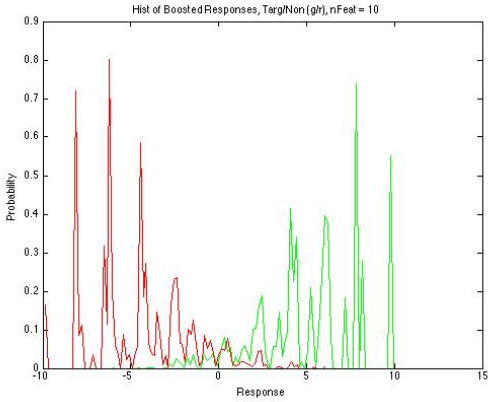
True/False Pos. Rates: 0.863294, 0.348619, θ_1: 1.224154, θ_2: -1, err: 0.231269
pFeat.sigma: 4, pFeat.deriv: 2, pFeat.theta: pi/8, pFeat.abs: false, x0: 10, y0: 15

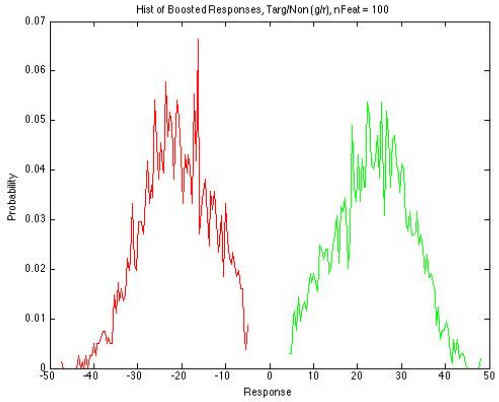


True/False Pos. Rates: 0.923077, 0.781136, θ_1: 2.335164, θ_2: 1, err: 0.390629
pFeat.sigma: 8, pFeat.deriv: 2, pFeat.theta: pi/16, pFeat.abs: true, x0: 10, y0: 15

2 AdaBoost

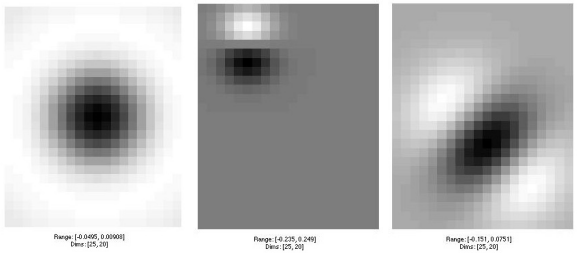
Using trainAdaGauss.m, we can train a series of weak classifiers. Below are 2 boosted response plots for m=10 and m=100:





Note that the positive and negative responses have larger seperation when m is larger.

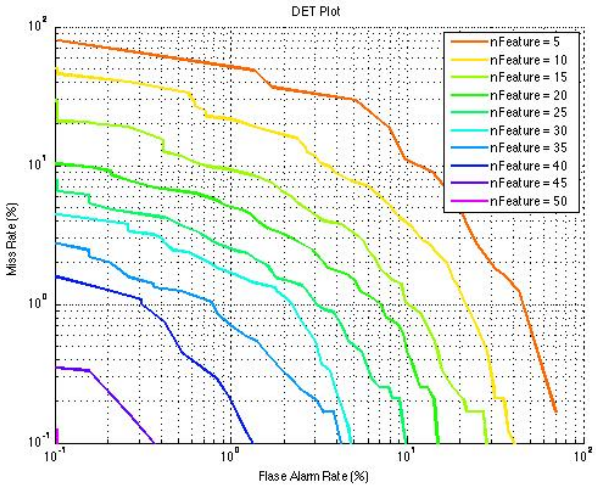
Below are some sample images for the selected feature:



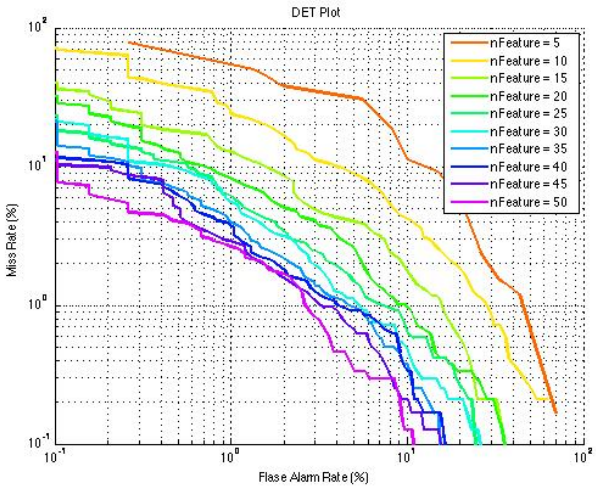
Left: normal Gaussian, middle: first-order Gaussian, right: second-order Gaussian

Note that if there is an edge in most training target images at a particular scale, sign and orientation, the expected selected feature would be a first or second order gaussian that is oriented in perpendicular direction with the edge (NOTE: second order gaussian is for edges that corresponse to line). The scale is likely choosen to be as close to the scale of the edge as possible. If the sign of the edge varies, likely the parity will change (see figure above, the black region and white region will be flipped depending on the sign). A brighter region for the trainning data will favor the white side of the gaussian and vice versa. Finally, if the training target images are relatively smooth in a particular region then the selected feature will likely be just a normal gaussian.

Below is DET curve for various value of M up to 50 for both train data and test data. My training actually did up to 100 features but the miss rate become so small that it is hard for the log scale curve to display.



trainSet DET curve

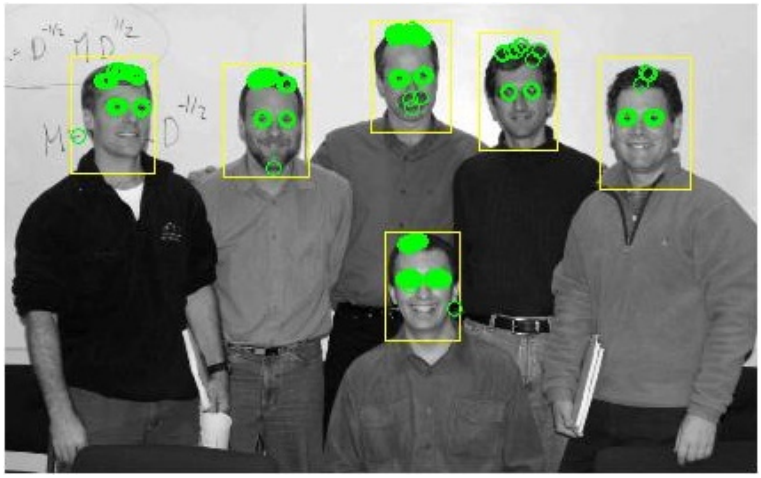


testSet DET curve

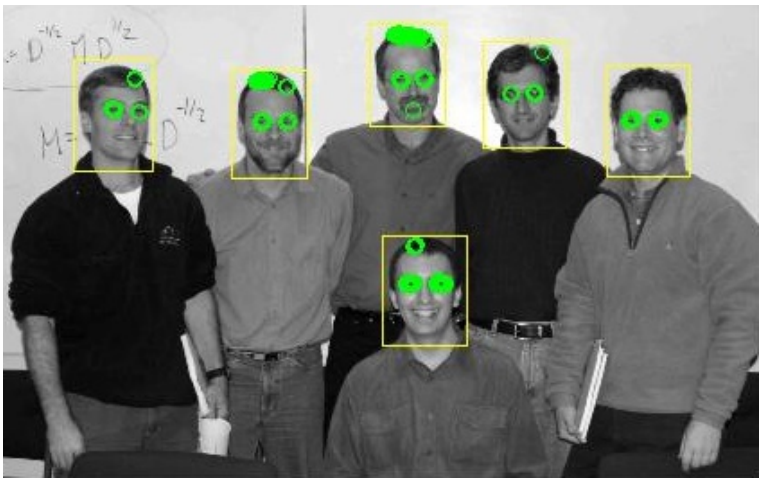
With larger number of features we see that our miss rate decreases. For trainSet we see that the miss rate reach almost to 0. Obviously, this is because we train our model with trainSet! For testSet, the miss rate decrease much slower which make sense because testSet contains images that we did not use for our trainning. Since the two sets contain eye images which suppose to look similar, we can see that the DET curve look similar and testSet have decent performance too.

3 Application to an Image Patch

Below is the result from running tryEyeDetector.m based on the method described in our assignment.



We can see that we detect the eyes but along with a lot of false positive results. With higher threshold t=10 we can obtain a better performance. We can still detect the eyes but still getting some small amount of false positive results. See blow:



Compare to Figure 1 in assignment 4 handout, we have more false positive recognitions. This make sense because the "Small Print" in the handout suggests that the trainSet we were given have smaller negative set.