MP6: Visual Face Detection

ECE 417 Fall 2017

Yuchen Liang

Zixu Zhang

November 29, 2017

1 Introduction

In this MP, we use Adaboost algorithm combined with Viola-Jones features to conduct face recognition of images. We trained 40 iterations of weak classifier use 126 labeled images, and constructed a strong classifier for face detection of the rest 42 images. In this report, we report unweighted error rates of both weak and strong classifiers and also weighted error rate of weak classifiers in each iteration.

2 Method

2.1 Adaboost Weak Classifier Training

First, we assume that we have N training images, and each of them have R two-class labeled rectangle Y. The general Adaboost Algorithm is shown as following. In training, we label rectangle with face as 1, and non-face rectangle as 0.

1) On line 25, we initialize a weight matrix W, such that $W \in \mathbb{R}^{N\times R}$, and each elements $w_{ij} = \frac{1}{NR}$.

2) At t th iteration, we can find a weak classifier $h_t(\mathbf{X}, p, \theta)$, where $\mathbf{X}$ are set of calculated features, p is the polarity, and θ is threshold. This is complished from line 44 to 50. Classification for j th rectangle in i the image is defined as

$$h_t(i,j) = \begin{cases} 1, & pX_{ij} < p\theta \\ 0, & pX_{ij} \geq p\theta \end{cases} \tag{1}$$

For t th weak classifier, we can find corresponding weighted classification error ϵ_t

$$\epsilon_t = \sum_{i=1}^{N}\sum_{j=1}^{R} w_{ij}\mathbb{1}_{h_{ij}\neq Y_{ij}} \tag{2}$$

$$\beta_t = \frac{\epsilon_t}{1-\epsilon_t} \tag{3}$$

3) We update our weight matrix W as

$$\tilde{w}_{ij} = \begin{cases} w_{ij}\beta_t & h_{ij} = Y_{ij} \\ w_{ij} & h_{ij} \neq Y_{ij} \end{cases} \tag{4}$$

Then, we normalize our updated weight matrix as

$$w_{ij} = \frac{\tilde{w}_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{R}\tilde{w}_{ij}} \tag{5}$$

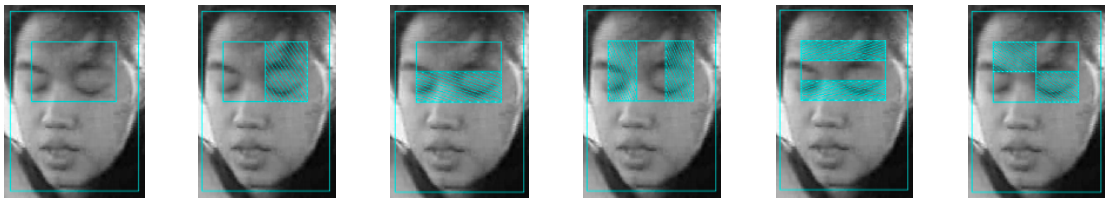
In this way, we increse the weight of wrong label, but reduce the weight of accurate label to ensure the accuracy improvement over iterations.

2.2 Scalar Features

In this MP, we use Viola-Jones features for face detection. In each grayscale image I, we are able to find its integral image II as

$$II(x,y) = \sum_{i\leq x, j\leq y} I(i,j) \tag{6}$$

In this way, it is efficient to calculate the sum of features of any rectangle subsection, by linear combination of integral image values of four corners. For each labeled rectangle in the set Y, we have identify its location in image with four parameters $[X_m, Y_m, W, H]$, where $[X_m, Y_m]$ is the pixel coordinate of its upper right corner, and W and H are its width and height. Similarly, we can identify the location of some subrectangle in side as $fr = [f_X, f_Y, f_W, f_H]$, such that the upper right corner of the subrectangle is $[X_m + Wf_X, Y_m + Hf_Y]$. The width and height of subrectangle is $[Wf_W, Hf_H]$.



(a) q=1 (b) q=2,v=0 (c) q=2,v=1 (d) q=3,v=0 (e) q=3,v=1 (f) q=4

Figure 1: Six Feature Orders

Figure 1b is the sum of the right half, minus the sum of the left half. Figure 1c is the sum of the bottom half, minus the sum of the top half. Figure 1d is the sum of the outer thirds, minus the sum of the middle third. Figure 1e is the sum of the outer thirds, minus the sum of the middle third. Figure 1f is the sum of the main diagonal quadrants, minus the sum of the off-diagonal quadrants. Scalar feature extraction of a given subrectangle is done in function `rectfeature`.

In order to find the optimal subrectangle, we divide each labeled rectangle into 6×6 grids and do exhaustic searches to find a combination of subrectangle and feature order with minimum weighted error ϵ_t that is used in weak classifier training. The exhaustic search are finished in line 29 to 57 in the code. After each iteration, we store the optimal subrectangle fr_t, threshold θ_t, polarity p_t and β_t of each iteration as trained classifier.

2.3 Strong Classifier and Testing

We first introduce a weight parameter α_t for each iteration t, and define it as

$$\alpha_t = -\ln \frac{\epsilon_t}{1 - \epsilon_t} = -\ln \beta_t \tag{7}$$

such that α_t is monotonically decreasing function of ϵ_t.

In testing, we say that rectangle with face is labeled as 1, and non-face rectangle is labeled as -1. Since we have 40 iterations of training, we obtained 40 optimal subrectangles for classification. First, we use our subrectangles at t th iteration to calculate scalar features of test images on line 77 by `rectfeature`. With trained threshold θ_t, and polarity p_t, we are able to use weak classifier to obtain the classification h_t of our test images by modified equation 1 as following.

$$h_t(i,j) = \begin{cases} 1, & pX_{ij} < p\theta \\ -1, & pX_{ij} \geq p\theta \end{cases} \tag{8}$$

After iterating through all trained classifier, we are able to construct a strong classifier H_t as following,

$$\tilde{H}_t(i,j) = \sum_{k=1}^{t} \alpha_k h_k(ij) \tag{9}$$

$$H_t(i,j) = \begin{cases} 1 & \tilde{H}_t(i,j) > 0 \\ -1 & \tilde{H}_t(i,j) < 0 \end{cases} \tag{10}$$

Therefore, we can calculated unweighted errors of both strong and weak classifier of testing images as

$$\epsilon_{weak} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{R} \mathbb{1}_{h_t(i,j) \neq Y_{ij}}}{NR} \tag{11}$$

$$\epsilon_{strong} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{R} \mathbb{1}_{H_t(i,j) \neq Y_{ij}}}{NR} \tag{12}$$

3 Result

We plotted unweighted errors of both strong and weak classifier and weighted error rates over 40 iterations in Figure 2

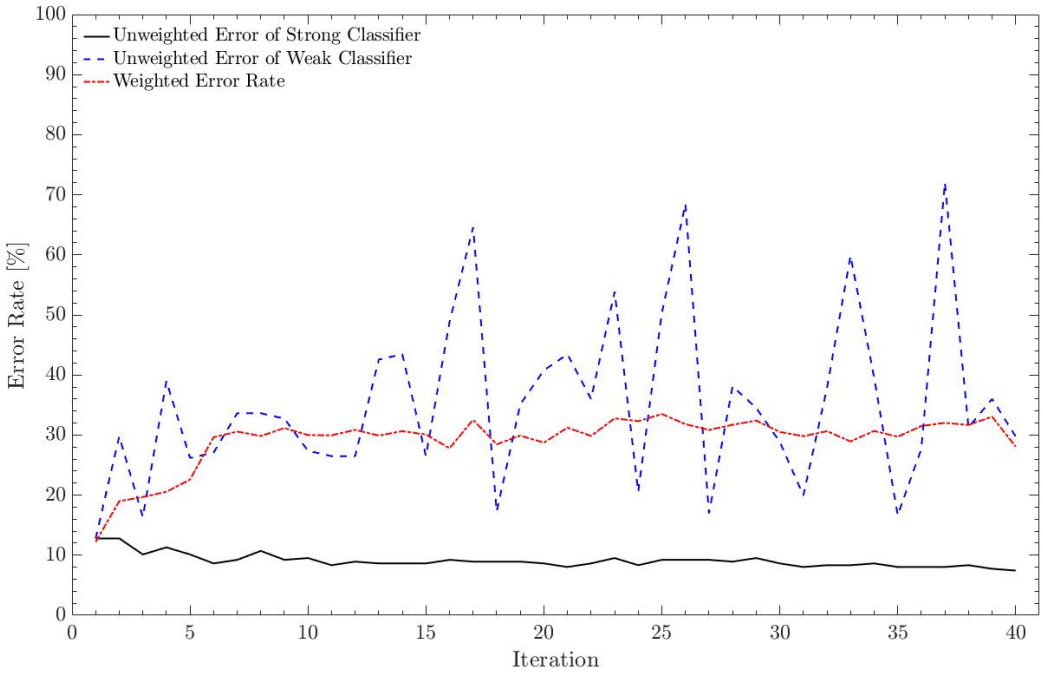


Figure 2: Facial Detection Errors Over 40 Iterations

4 Discussion

Over the 40 training iterations, the weighted error rate increases in the first 6 iterations and vacillates around 31 percent in later iterations. This is a result of the fact that we update the weight matrix W after every iteration by reducing the weight of those correctly labeled features while raising that of the others. This also results in the non-convergence and the increasing error rate of the weak classifiers.

On the other hand, the strong classifier using Adaboost has a gradually decreasing error rate that converges to around 9 percent, an error rate much lower than that from any of the weak classifiers. Due to the way the weight matrix W is updated, the weak classifier in the next iteration is designed to fix the mistake made by the current one. The alpha coefficients are designed to be larger for smaller error rate, and vice versa. So the whole strong classifier takes the advantage of all weak classifiers and output the best classification result.