MP6: Image Object Detection

Group: Akhil Alapaty & Ojus Deshmukh

I. Introduction.

For this MP, we are dealing with the Viola-Jones feature vector. The Viola-Jones framework was designed for face detection: being able to detect faces from non-faces. Although this feature requires each face to be perfectly front-facing (can be skewed by the lightest tilt), it gives significantly more data and context than raw pixels, as it is a real-time algorithm with a very high detection rate. The computation of this feature is comprised of many steps. First, the algorithm relies on detecting Haar features in the given images; Haar features refer to the common features shared by all human faces. These features are defined as “rectangular filters”, which serve as the differences between the sums of the pixels in each of the adjacent rectangles. This resulting feature is also known as a 2-rectangle feature. The computation of these features can be done by using Integral Images. After using the AdaBoost training algorithm to select the best features and to test/train the respective classifiers on those features, the Cascaded classifier works by setting multiple classifiers for each of the features.

As mentioned before, the learning algorithm used in this MP is the AdaBoost training algorithm. It works by utilizing numerous learning algorithms to build a stronger learning algorithm. For example, it chooses a base algorithm such as the decision trees algorithm, which is improved by taking into account each of the false classifications seen in the training set. So, it is a combination of a classifiers with weak weights in order to create a strong classifier. For MP6, we use the FACE features provided to us in allrects.txt, each of which have their corresponding Xmin, Ymin, Width, and Height data.

II. Methods

One of our critical transformations was creating the classifier using the Adaboost algorithm. We start with 126 training images. Adaboost creates 40 weak classifiers’ outputs to create one final strong classifier. Since each image has 8 training rectangles, we have 126*8 weightings that start out equal, then change as we run adaboost_learn.m. To determine the weightings, we search each possible one-dimensional feature to get the lowest possible classification error. This happens in the 7 for loops starting at line 28. We use this error probability to give each of

the 40 weak classifiers a weight, which creates our final strong classifier. All of this was done in the file adaboost_learn.m.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,

     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:

     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

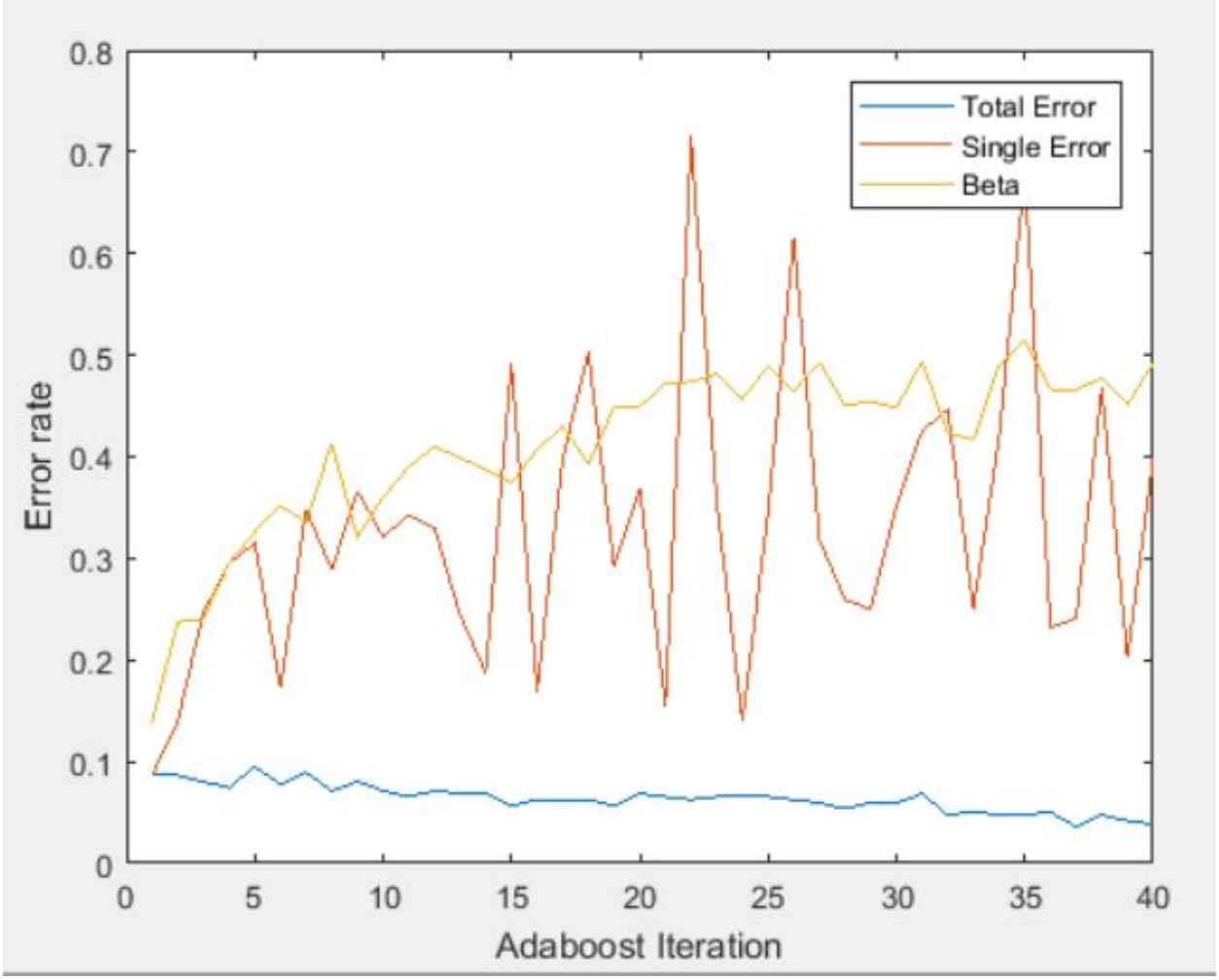
where $\alpha_t = \log \frac{1}{\beta_t}$

Our second critical transform was testing the rectangles to find out which were faces and which were background rectangles. To do this, we took each of our 40 weak classifiers that we generated from adaboost_learn.m and generated a +/- 1 label for each rectangle. We multiplied each label by its classifier's alpha, given here:

$$\alpha_t = -\log(\frac{P_E}{1 - P_E})$$

where P is the error probability. If the resulting product was negative or equal to zero, we classified that rectangle as a non-face rectangle. If the product was positive, we classified that rectangle as a face rectangle. The reason we multiplied by alpha, was because we wanted alpha to be a monotonically decreasing function of P, that is, strictly decreasing. Of course, the classifier with the lowest P is the best one, but we wanted the other, slightly less accurate classifiers to fill in the blanks that the accurate classifier couldn't. This log function gives them a smaller weight accordingly.

III. Results



IV. Discussion

As seen in the graph, we see that the Single Classifier and Beta increase in error as the number of AdaBoost iterations increase. However, we also see that Total Error consistently decreases with each iteration since the AdaBoost algorithm effectively fixes the errors with each training sample and builds a stronger classifier.