ECE 661: Homework 8
Due December 2nd

Problem:

In the last homework you experimented with object recognition, identifying which person is visible in a set of face images. A related problem is object detection in which you want to identify instances of a general object class (e.g. people, cars, dogs, etc.) in real world images. We can think of this as a two-class classification problem, e.g. cars vs non-cars. In the last homework you used a classifier which consisted of first reducing the dimensionality of the data and then doing nearest-neighbor. In this homework you will study a much more powerful classifier which is formed by cascading Adaboost classifiers.

In class you have already studied the Adaboost classifier. A single Adaboost classifier consists of a weighted sum of many weak classifiers. The power of Adaboost lies in how these individual weak classifiers are trained. The weak classifiers are trained on weighted data, meaning that not all data samples are treated the same: the penalty associated with misclassifying a data sample is equal to its weight. Initially the weights are evenly distributed across all samples. After each weak classifier is trained, the weight on misclassified samples is increased while the weight on correcly classified samples is decreased. In this way the overall classification error can be driven extremely low. As a simple toy to play around with Adaboost you may find this online applet interesting: http://cseweb.ucsd.edu/~yfreund/adaboost/

For this homework we will consider the situation where each weak classifier is a threshold on a single feature. In this context, we may think of Adaboost as performing automatic feature selection at each stage. To get good performance we need a very large pool of features. The features do not need to be independent and in fact it is perferable to have some redundancy built in. Following the example of the Viola and Jones face detector, we will use haar-like features.

Haar-like features may be thought of as smoothed derivatives at different scales. A feature is given by the sum of pixel intensities in one rectangle minus the sum in a neighboring rectangle of the same size (note that this is more limited than the features in Viola and Jones). This reduces to types of features with the rectangles either sharing a vertical edge or a horizontal edge (the choice of which rectangle is positive and which is negative is not important as long as you are consistent). This leads to a huge number of features which can be computed very efficiently as will be described in the next section.

Each Adaboost classifier should be trained following the steps in Table 1 on page 142 of the Viola and Jones paper. The weak classifier h is described at the bottom of the right column on page 141. For the final strong classifier $C(x)$, instead of using

$$\frac{1}{2}\sum_{t=1}^{T}\alpha_t$$

as the threshold you should set the threshold so that all of the positive training samples are correctly classified (i.e. there are no false negatives). To determine the correct number of weak classifiers T to use, keep adding features until the final strong classifier prunes away at least 50% of the negative samples and then stop.

While a single Adaboost classifier can be quite strong, a cascade of Adaboost classifiers is needed in order to get the extremely low false positive rates required for object detection. The key idea in a cascade is that a sample is classified as positive only if every classifier in the cascade classifies it as positive. In training the classifier, only training samples which are classified as positive in all of the preceding stages (i.e. both true positives and false positives) are used. In this way, increasingly difficult false positives are pruned away at each stage resulting in an overall classifier which has an extremely low false positive rate. The number of cascades to use is determined by how low the false positive rate on the training data should be. For this homework you should keep adding stages until all of the samples are correctly classified (this should require less than 10 stages on the provided data set).

A link to the data for this homework is given on the homework page on the website. You will need a username and password which I will e-mail out. Just like the last homework, the data is split into a train and test folder for use in training and testing respectively. Within each folder is a positive folder with images of cars and a negative folder with images of non-cars. **This data comes from actual research and is only being shared with students of ECE661 for the purpose of completing this homework. You are not permitted to share this data with anyone else. After completing the homework, you must delete the data.**

Computing Haar-Like Features:

The image patches you will be attempting to classify are 20×40 pixels. Even using just two types of haar-like features this will still lead to a total of 166,000 features per patch. This is a huge feature space and so some care is required in the implementation in order to limit the memory requirements and achieve a reasonable running time.

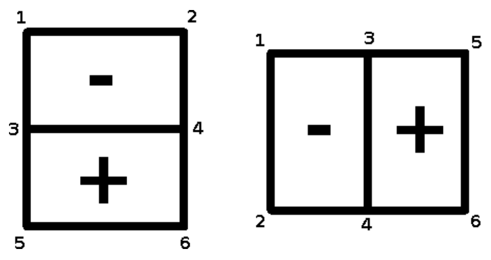
To efficiently compute the haar-like features, first compute the integral image of each input I which is given by

$$S(i,j) = \sum_{x<i} \sum_{y<j} I(x,y)$$

Note that S should start with a row and column of zeros and thus should be 21×41. Each haar-like feature can then be expressed as the weighted sum of 6 entries in S of the form

$$f = -S(x_1,y_1) + S(x_2,y_2) + 2S(x_3,y_3) - 2S(x_4,y_4) - S(x_5,y_5) + S(x_6,y_6)$$

with the locations of coordinates 1-6 for vertical and horizontal features shown in the images below.



Note that you will not be able to compute all of the haar-like features for all of the images at once (that would require around 3GB of RAM just to store the result). However, to the extent possible you should try to work on batches of features at a time (a few thousand features can be applied to all of the images and still not require too much RAM).

Even with efficient feature extraction, training the classifier will take a long time (i.e. hours). Definitely do not wait until the last minute to start this homework or you'll never finish.

Solution:

You should turn in a report in pdf format of your homework solution using electronic turn-in. The report should include:

1. A brief overview of the classifier including the full training procedure.
2. A plot showing the false positive and false negative rate after the first k stages of the cascade as a function of k
   (a) Compute the false positive rate using
   $$fp = \frac{\text{\# of misclassified negative test images}}{\text{\# of negative test images}}$$
   Is the false positive rate low enough for object detection?
   (b) Compute the false negative rate using
   $$fn = \frac{\text{\# of misclassified positive test images}}{\text{\# of positive test images}}$$
3. Your source code (you are strongly encouraged to use MATLAB).

Finally, this homework is new for this year and is pretty challenging so there may be some kinks to work out. Just do the best you can and feel free to ask questions.