**Recitation 9. November 19, 2004**

6.034 - Artificial Intelligence

**The Boosting Algorithm AdaBoost**

**AdaBoost Algorithm.**

Given training examples $(x_1, y_1), \ldots, (x_m, y_m)$ such that $x_i \in X, y_i \in Y = \{-1, +1\}$.
Initialize $D_1(i) = 1/m$. ($D_t(i)$ represents how much weight is given to example $i$ on iteration $t$.)
For $t = 1, \ldots, T$:

1. Train weak learner using distribution $D_t$
2. Get weak classifier $h_t : X \rightarrow Y$ ($h_t$ can be an ID tree, a NN-based classifier, ...)
3. Compute the error of the classifier $h_t$:

$$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i]$$

   and use the error to compute $\alpha_t \in R$:

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$$

   ($\alpha_t$ represents the weight on each classifier.)

4. Update

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

   where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution)

Output the final classifier to be a weighted majority vote of the $T$ base classifiers:

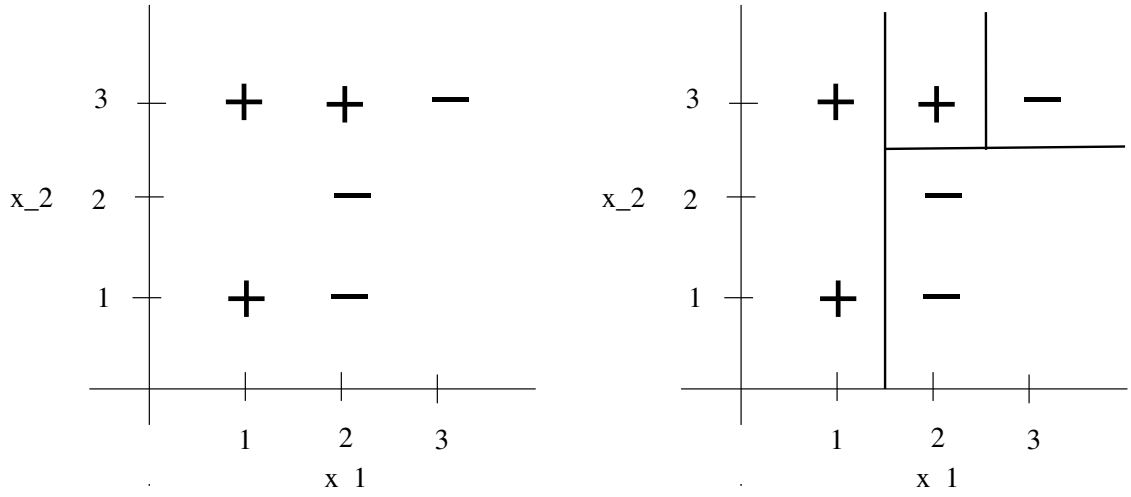
$$H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

Slogan: "T heads are better than 1."
Important properties of Adaboost:

- Integrates disparate classifiers together
- Theoretical bounds – adding a new classifier can't hurt
- Easy to program, doesn't get stuck in local minima on its own
- Sensitive to outliers, prone to overfitting

**Problem 1.**

Suppose we had the following dataset on the left-hand side below:




The ID tree algorithm classifier line is shown on the right-hand side above. Let's run 3 steps on AdaBoost on the same dataset, using axis-parallel weak classifiers, or in other words, single-test ID trees as the weak classifiers (also known as decision stumps).

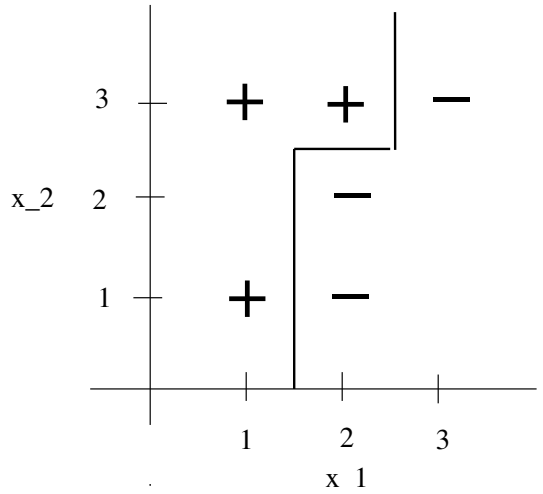
There are only eight possible base classifiers based on the four possible decision boundaries. The four possible tests are:

1. $x_1 \geq 1.5$
2. $x_1 \geq 2.5$
3. $x_2 \geq 1.5$
4. $x_2 \geq 2.5$

Here is the table to build:

| $i : (x_1, x_2); y_i$ | $D_1(i)$ | $D_2(i)$ | $D_3(i)$ |
|---|---|---|---|
| $1 : (1,1); +1$ | $1/6$ | $1/10$ | $1/16$ |
| $2 : (1,3); +1$ | $1/6$ | $1/10$ | $1/16$ |
| $3 : (2,3); +1$ | $1/6$ | $1/2$ | $5/16$ |
| $4 : (2,1); -1$ | $1/6$ | $1/10$ | $1/4$ |
| $5 : (2,2); -1$ | $1/6$ | $1/10$ | $1/4$ |
| $6 : (3,3); -1$ | $1/6$ | $1/10$ | $1/16$ |
| Weak classifier $h_t$ | $h_1 =$<br>$+1$ if $x_1 \leq 1.5$<br>$-1$ if $x_1 > 1.5$ | $h_2 =$<br>$+1$ if $x_1 \leq 2.5$<br>$-1$ if $x_1 > 2.5$ | $h_3 =$<br>$+1$ if $x_2 \geq 2.5$<br>$-1$ if $x_2 < 2.5$ |
| Error $\epsilon_t$ | $\epsilon_1 = 1/6$ | $\epsilon_2 = 1/5$ | $\epsilon_3 = 1/8$ |
| Weights $\alpha_t$ | $\alpha_1 = (1/2)\ln(5)$ | $\alpha_2 = \ln(2)$ | $\alpha_3 = (1/2)\ln(7)$ |

What is the resulting classifier of AdaBoost? In others words, how does it treat new data? Draw AdaBoost's decision boundary, and compare it to the decision boundary of the ID tree.




Here are the calculations used to fill out the table. To save space and for simplicity, we denote the two-class disorder (entropy) function by $H(p) = -p\log_2(p) - (1-p)\log_2(1-p)$. (Note that the $H$ is symmetric; that is, $H(p) = H(1-p)$.) For consistency, we use the weighted proportion of the positive examples when invoking $H$ (*ie*, if $p$ is the weighted proportion of positive examples for the branch of the test, we use $H(p)$ to denote the disorder for the branch).

| Round | Test | Average disorder |
|---|---|---|
| 1 | **1** | $\mathbf{(2/6) * 0 + (4/6) * H(1/4) \approx 0.52}$ |
| | 2 | $(5/6) * H(3/5) + (1/6) * 0 \approx 0.81$ |
| | 3 | $(2/6) * H(1/2) + (4/6) * H(2/4) = 1$ |
| | 4 | $(3/6) * H(1/3) + (3/6) * H(2/3) \approx 0.92$ |
| 2 | 1 | $(2/10) * 0 + (8/10) * H(5/8) \approx 0.76$ |
| | **2** | $\mathbf{(9/10) * H(7/9) + (1/10) * 0 \approx 0.688}$ |
| | 3 | $(2/10) * H(1/2) + (8/10) * H(6/8) \approx 0.85$ |
| | 4 | $(3/10) * H(1/3) + (7/10) * H(6/7) \approx 0.690$ |
| 3 | 1 | $(2/16) * 0 + (14/16) * H(5/14) \approx 0.82$ |
| | 2 | $(15/16) * H(7/15) + (1/16) * 0 \approx 0.93$ |
| | 3 | $(5/16) * H(1/5) + (11/16) * H(6/11) \approx 0.93$ |
| | **4** | $\mathbf{(9/16) * H(1/9) + (7/16) * H(6/7) \approx 0.54}$ |

Here is how the new weights of the samples are computed.

| $i : (x_1, x_2); y_i$ | $D_1(i)$ | $D_2(i)$ | $D_3(i)$ |
|---|---|---|---|
| $1 : (1,1); +1$ | 1/6 | $\propto (1/6)(1/\sqrt{5})$; 1/10 | $\propto (1/10)(1/2)$; 1/16 |
| $2 : (1,3); +1$ | 1/6 | $\propto (1/6)(1/\sqrt{5})$; 1/10 | $\propto (1/10)(1/2)$; 1/16 |
| $3 : (2,3); +1$ | 1/6 | $\propto (1/6)(\sqrt{5})$; 1/2 | $\propto (1/2)(1/2)$; 5/16 |
| $4 : (2,1); -1$ | 1/6 | $\propto (1/6)(1/\sqrt{5})$; 1/10 | $\propto (1/10)(2)$; 1/4 |
| $5 : (2,2); -1$ | 1/6 | $\propto (1/6)(1/\sqrt{5})$; 1/10 | $\propto (1/10)(2)$; 1/4 |
| $6 : (3,3); -1$ | 1/6 | $\propto (1/6)(1/\sqrt{5})$; 1/10 | $\propto (1/10)(1/2)$; 1/16 |

Here is how the decision boundary is computed.

| $(x_1, x_2)$ | $h_1$ | $h_2$ | $h_3$ | $H$ |
|---|---|---|---|---|
| $(1,1)$ | $+1$ | $+1$ | $-1$ | $+1$ |
| $(1,2)$ | $+1$ | $+1$ | $-1$ | $+1$ |
| $(1,3)$ | $+1$ | $+1$ | $+1$ | $+1$ |
| $(2,1)$ | $-1$ | $+1$ | $-1$ | $-1$ |
| $(2,2)$ | $-1$ | $+1$ | $-1$ | $-1$ |
| $(2,3)$ | $-1$ | $+1$ | $+1$ | $+1$ |
| $(3,1)$ | $-1$ | $-1$ | $-1$ | $-1$ |
| $(3,2)$ | $-1$ | $-1$ | $-1$ | $-1$ |
| $(3,3)$ | $-1$ | $-1$ | $+1$ | $-1$ |

Each entry is computed by evaluating each weak classifier $h_t$ on the input, then computing the $\alpha_t$-weighted majority of the outputs of the weak classifiers.

## Problem 2.

Now that you have some experience running through the algorithm, let's explore some questions:

1. How does the weight $\alpha_t$ given to classifier $h_t$ relate to the performance of $h_t$ as a function of the error $\epsilon_t$?

   *Answer:* The lower the error $\epsilon_t$, the better the classifiers $h_t$ is on the (weighted) training data, the larger the weight $\alpha_t$ we give to the classifier output when classifying new examples.

2. How does the error of the classifier $\epsilon_t$ affect the new $D_t(i)$ on the samples?

   *Answer:* The lower the $\epsilon_t$, the better the classifier $h_t$ classifies the (weighted) training examples, hence the larger the increase on the weight of the samples that it classifies incorrectly and similarly the larger the decrease on those that it classifies correctly. More generally, the smaller the error, the more significant the change in the weights on the samples.

   Note that this dependence can be seen indirectly in the AdaBoost algorithm from the weight of the classifier $\alpha_t$. The lower $\epsilon_t$, the larger $\alpha_t$, the better $h_t$ is on the (weighted) training data.

3. How can we adapt the distortion-based ID tree learning algorithm to handle weighted samples?

   *Answer:* All we need to do is to modify the distortion measure as follows. Consider a particular test to add to the ID tree and the particular examples involved in the test. For each branch generated by the test, the distortion (or entropy) is measured using the *average weight* of the examples of each class as opposed to just the *number* of examples of each class. The average distortion for the test is then the weighted average distortion of all the branches.

   For instance, assume we want to classify examples into positive and negative and each test leads to only two branches, call them left and right. Let $p_1$ be the total weight of the positive examples that fall in the left branch. Similarly, let $n_1$ be the total weight of the negative examples that fall in the left branch. Note that the total weight of the examples that fall in the left branch is $w_1 = p_1 + n_1$. The distortion for the left branch is then given by $d_1 = -(p_1/w_1)\log_2(p_1/w_1) - (n_1/w_1)\log_2(n_1/w_1)$. We can obtain the distortion for the right branch, which we denote by $d_2$, similarly. Note that $w = w_1 + w_2$ is the total weight of all the examples available to the test. Now the average distortion for the test is $(w_1/w)d_1 + (w_2/w)d_2$.

4. How does AdaBoost end up treating outliers?

   *Answer:* AdaBoost can help us identify outliers since those examples are the hardest to classify and therefore their weight is likely to keep increasing as we add more weak classifiers. At the same time, the theoretical bound on the training error implies that as we increase the number of base/weak classifiers, the final classifier produced by AdaBoost will classify all the training examples. This means that the outliers will eventually be "correctly" classified from the standpoint of the training data. Yet, as expected, this might lead to overfitting.

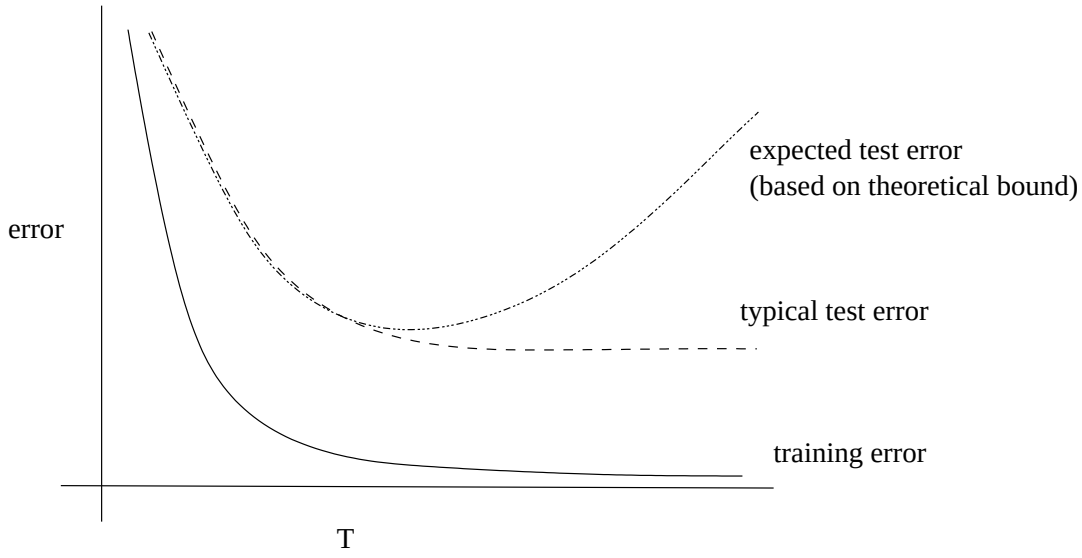
5. Why is it not the case that the new classifiers "clash" with the old classifiers on the training data?

   *Answer:* The intuition is that, by varying the weight on the examples, the new weak classifiers are trained to perform well on different sets of examples than those for which the older weak

classifiers were trained on. A similar intuition is that at the time of classifying new examples, those classifiers that are not trained to perform well in such examples will cancel each other out and only those that are well trained for such examples will prevail, so to speak, thus leading to a weighted majority for the correct label.

6. Draw a picture of the training error, theoretical bound on the true error, and the typical test error curve:

   *Answer:*




7. Do we expect the error of new weak classifiers to increase or decrease with the number of rounds? Why?

   *Answer:* We expect the error of the weak classifiers to increase in general since they have to perform well in those examples for which the weak classifiers found earlier did not perform well. In general, those examples will have a lot of weight yet they will also be the hardest to classify correctly.