Boosting

The main idea in Boosting is that we are trying to combine (or ensemble) of "weak" classifiers (classifiers that underfit the data) h(x) into a single strong classifier H(x).

$$H(\vec{x}) = sign(\alpha_1 h_1(\vec{x}) + \alpha_2 h_2(\vec{x}) + \ldots + \alpha_s h_s(\vec{x}))$$
$$H(\vec{x}) = sign(\sum_i^s \alpha_i h_i(\vec{x}))$$

where:

$$H(\vec{x}) \in \{-1,+1\} \quad h_i(\vec{x}) = \{-1,+1\}$$

Each data point is weighed. w_i for $i \in 1\ldots n$. **Weights** are like probabilities, (0, 1], with $\sum_i^n w_i = 1$. But weights are never 0; this implies that all data points will have some vote at all times.

Decision stump weights:

$$\alpha_s = \ln\left(\frac{1-E^s}{E^s}\right)^{\frac{1}{2}} = \frac{1}{2}\ln\left(\frac{1-E^s}{E^s}\right)$$

Definition of Errors:

$$E^s = \sum_{incorrect} w_i \qquad (1-E^s) = \sum_{correct} w_i$$

In Boosting we always pick stumps with errors < 1/2. Because stumps with errors > 1/2 can always be flipped. Stumps with error = 1/2 are useless because they are no better than flipping a fair coin.

$E^s < \frac{1}{2}$ and $1 - E^s > \frac{1}{2}$ so $E < (1-E)$ => $\frac{(1-E)}{E} > 1$ Therefore: $\alpha_s > 0$.

Adaboost Algorithm

Input: Training data $(\vec{x_1}, y_1) \ldots (\vec{x_n}, y_n)$

1. Initialize $w_i^1 = \frac{1}{n} \quad \forall\, i \in (1\ldots n)$ a weight for each data point.
2. For s = 1 ... T:
   a. Train base learner using distribution w^s on training data. Get a base (stump) classifier $h_s(\vec{x})$ that achieves the lowest E_s (error). [Note in examples that we do in class, $h_s(\vec{x})$ are picked from a set of predefined stumps, this procedure of "picking" the best stump is the same as "training".]
   b. Compute the stump weight: $\alpha_s = \frac{1}{2}\ln\frac{(1-E_s)}{E_s}$
   c. Update weights (there are three ways to do this):
      - Original: $w_i^{s+1} = \frac{e^{-\alpha s}}{N^s} \cdot w_i^s$ (correct pts.) $w_i^{s+1} = \frac{e^{-\alpha s}}{N^s} \cdot w_i^s$ (incorrect pts.)
      - OR more human-friendly: $w_i^{s+1} = \left[\frac{1}{2} \cdot \frac{1}{1-E^s}\right] \cdot w_i^s$ (correct pts.) $w_i^{s+1} = \left[\frac{1}{2} \cdot \frac{1}{E^s}\right] \cdot w_i^s$ (incorrect pts.) (see derivation below)
      - OR use **numerator-denominator method** (see below)
1. Output the final classifier: $H(\vec{x}) = sign(\sum_s \alpha_s h_s(\vec{x}))$

Possible Termination conditions:
1. Stop after T rounds (we manually set some T.)
2. Stop after H(x) (final classifier) has error = 0 on training data or < some error threshold.
3. Stop when you can't find any more stumps h(x) where weighted error is < 0.5. (i.e. All stumps have E = 0.5).

The Numerator-Denominator method

A calculator-free method for finding weight updates quickly

<u>Replace the Weight Update Step 1c above with these steps.</u>

1. **Write** all weights in the form of $w_i = \frac{n_i}{d}$

where the denominator *d* is the same to all weights.

2. **Circle** the data points that are incorrectly classified.

3. Compute the **new denominator** for (the circled) incorrectly classified points:

$$d'_{incorrect} = 2 \cdot \left(\sum_{incorrect} n_i\right)$$

which is sum of all the incorrect numerators times two.
Compute the new denominator for (uncircled) correct points:

$$d'_{correct} = 2 \cdot \left(\sum_{correct} n_i\right)$$

sum of all the correct numerators times two.

4. **New weights** are the old numerator divided by the updated denominators found in step 3.

$w'_i = \frac{n_i}{d'_{incorrect}}$ if incorrect

$w'_i = \frac{n_i}{d'_{correct}}$ if correct.

5. Adjust all the numerators and denominators such that the denominator is again the same for all weights. Optional: Check and make sure correct weights add up to 1/2, and incorrect weights also add up to 1/2.

A Shortcut on computing the output of H(x).

Quizzes often ask you for the Error of the final **H(x)** ensemble classifier on the training data.
Here is a quick way to compute the output of H(x) without calculating logarithms.
Step 1: compute the sign of each of stump h(x) on the given data point.
Step 2: compute products of the log arguments of the +ve stumps and -ve stump.

If $\prod_{+} \frac{1-E_s}{E_s} > \prod_{-} \frac{1-E_s}{E} \rightarrow +$ If $\prod_{+} \frac{1-E_s}{E_s} < \prod_{-} \frac{1-E_s}{E} \rightarrow -$

Example: suppose $H(\vec{x}) = \frac{1}{2} sign(\ln(5) \cdot h_1(\vec{x})) + \ln(2) \cdot h_2(\vec{x})) + \ln(2) \cdot h_3(\vec{x}))$

if $h_1(x)$ is + and $h_2(x)$ is + and $h_3(x)$ is -ve

(5 * 2) > 2 H(x) should output +ve

if $h_1(x)$ is + and $h_2(x)$ is - and $h_3(x)$ is -ve

5 > (2 * 2) H(x) should output +ve.

Step 3: Once you've computed all of the H(x) output values on the training data points, count the number of case where H(x) disagrees with the true output. That is the error.

FAQ

Dear TA, how do I determine if a stump will "never" be used (such as for part 1.A of 2006 Q4)?

Test stumps that are never used are ones that make more errors than some pre-existing test stump. In other words, if the set of mistakes stump X makes is a **superset** of errors stump Y makes, then Error(X) > Error(Y) is always true, no matter what weight distributions we use. Hence we will always chose Y over X because it makes less errors. So X will never be used!

Here is the answer to problem 1A from the 2006 Q4 with explanation. Setup: We are given the tests and the mistakes they make on the training examples, and we are asked to cross out the tests that are **never** used.

Test	Misclassified examples	**Never** used? Reason?
TRUE	1,2,3,5	Yes, superset of G=Y or U!=N
FALSE	4,6	Yes, superset of U=M
C=Y	1,6	No,
C=N	2,3,4,5	Yes, superset of G=Y or U=M
U=Y	1,2,3,6	Yes, superset of U!=N
U!=Y	4,5	Yes, superset of G=Y or U=M
U=N	4,5,6	Yes, superset of G=Y or U=M
U!=N	1,2,3	No,
U=M	4	No,
U!=M	1,2,3,5,6	Yes, superset of G=Y, C=Y or U!=N
G=Y	5	No,
G=N	1,2,3,4,6	Yes, superset of U=M, C=Y or U!=N

Food For thought:
Suppose we were to come up with a strong classifier that is a uniform combination of stumps (equal weights).
Q: How many mis-classifications would the following classifier commit?

H(x) = h(FALSE) + h(C=Y) + h(U!=Y)

A: Combining the misclassification sets of the stumps: {4, 6}, {1, 6}, {4, 5}
Points 1, 5 will be misclassified by 1 stump and correctly classified by 2 stumps.
So H(x) will be correct on 1, 5.
Points 4, 6 will be misclassified by 2 stumps and correctly classified by 1 stump.
So H(x) will misclassify 4, 6.
Therefore the points H(x) will mis-classify will be {4, 6}
How many misclassifications would H(x) = h(U=M) + h(G=Y) + h(C=Y) make?

(Optional 1) Derivation of the human-friendly weight update equations

Here is how the original weight update equations for Adaboost was derived into the more human friendly version. The original Adaboost weight update equations were

$w_i^{s+1} = \frac{w_i^s}{N^s} \cdot e^{-\alpha_s}$ For correctly classified samples (we ***reduce*** their weight)

$w_i^{s+1} = \frac{w_i^s}{N^s} \cdot e^{+\alpha_s}$ For incorrectly classified samples (we ***increase*** their weight)

Plug in alphas and redefine the exponential terms in terms of errors E:

$w_i^{s+1} = \frac{w_i^s}{N^s}\sqrt{\frac{E^s}{1-E^s}}$ For correctly classified examples.

$w_i^{s+1} = \frac{w_i^s}{N^s}\sqrt{\frac{1-E^s}{E^s}}$ For incorrectly classified examples.

Next, plug in the normalization factor (derived in Prof. Winston's handout)

$$N^s = 2\sqrt{E^s(1-E^s)}$$

Then simplifying gives us the:

$$w_i^{s+1} = \left[\frac{1}{2} \cdot \frac{1}{1-E^s}\right] \cdot w_i^s$$ for correctly classified examples.

$$w_i^{s+1} = \left[\frac{1}{2} \cdot \frac{1}{E^s}\right] \cdot w_i^s$$ for incorrectly classified samples.

To check your answers. Always sum up weights (for correct or wrong weights), they must each add up to 1/2!

$$\sum_{correct} w_i^{s+1} = \frac{1}{2} \cdot \frac{\sum_{correct} w_i^s}{1-E^s} = \frac{1}{2}$$

$$\sum_{incorrect} w_i^{s+1} = \frac{1}{2} \cdot \frac{\sum_{incorrect} w_i^s}{E^s} = \frac{1}{2}$$

(Optional 2) Proof of correctness of numerator-denominator method

In this proof we use the short hand: $w_i' \equiv w_i^{s+1}$

$$w_i = \frac{n_i}{d} \qquad w_i' = \frac{n_i}{d'}$$

In the procedure, we keep the numerator constant, only the denominator is updated during the weight update step from d to d'.
Starting from the weight update equations (for correct points):

1. $w_i' = \frac{1}{2} \cdot w_i \cdot \frac{1}{1-E^s}$	2. $\frac{n_i}{d'} = \frac{1}{2} \cdot \frac{n_i}{d} \cdot \frac{1}{1-E^s}$
3. $\frac{1}{d'} = \frac{1}{2} \cdot \frac{1}{d} \cdot \frac{1}{\sum_{j \in correct} \frac{n_j}{d}}$	4. $\frac{1}{d'} = \frac{1}{2} \cdot \frac{1}{d} \cdot \frac{d}{\sum_{j \in correct} n_j}$
5. $d'_{correct} = 2 \sum_{j \in correct} n_j$	

The proof for incorrect points will yield the same result. This shows that the denominator update rule used in step 3 can be derived directly from the weight update equations so it is correct.