AdaBoost Algorithm Overview

Input

A set of n training examples, each with p features (represented by matrix $\boldsymbol{X}$), and a vector of labels $\boldsymbol{y}$ where each $y \in \{-1, 1\}$ (binary classification). We also need to choose a family of classifiers (i.e. decision stumps) that will work with *weighted* training examples, and a number of iterations T.

Initialization

- Assign uniform weights to all training data points: $w_i^{(1)} = \frac{1}{n}$ for $i = 1, 2, \cdots, n$. Note that we require the weights to sum to 1.

Adaptive Procedure

For $t = 1, 2, \cdots T$, use the following procedure to find a new classifier and update the weights on the training examples:

(a) Fit a classifier to the weighted training set. We will call this classifier $h^{(t)}(\boldsymbol{x})$.

(b) Compute weighted classification *error* on the training set:

$$\epsilon_t = \sum_{i=1}^{n} w_i^{(t)} \mathbb{1}\left(y_i \neq h^{(t)}(\boldsymbol{x}_i)\right)$$

Note that since the weights sum to 1, $0 \leq \epsilon_t \leq 1$. However, since we are in a binary classification scenario, we should never have an error greater than 0.5.

(c) Compute the *score* of the classifier:

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

The score is 0 when $\epsilon_t = \frac{1}{2}$ (random guessing). As $\epsilon_t \to 0$, $\alpha_t \to \infty$ (i.e. a very good classifier).

(d) Using the score, update the weights on all the training examples. These are the weights we will use for the next iteration:

$$w_i^{(t+1)} = c_t w_i^{(t)} \exp\left(-y_i \alpha_t h^{(t)}(\boldsymbol{x}_i)\right),$$

where c_t is normalizer to ensure all the weights add to 1:

$$c_t = \frac{1}{\sum_{i=1}^{n} w_i^{(t)} \exp\left(-y_i \alpha_t h^{(t)}(\boldsymbol{x}_i)\right)}$$

If the data point is classified correctly, y_i and $h^{(t)}(\boldsymbol{x}_i)$ have the same sign and thus the previous weight is multiplied by $e^{-\alpha_t}$. Since α_t is positive, this quantity is less than 1, so the weight is *decreased*. If the data point is classified incorrectly, y_i and $h^{(t)}(\boldsymbol{x}_i)$ have the opposite sign and thus the previous weight is multiplied by $e^{-\alpha_t}$. This is greater than 1, so the weight is *increased*.

Testing

For each test data point $\boldsymbol{x}$, classify it as 1 or -1 using each classifier. Then weight the predictions by the score of the classifier during training. So our final prediction/hypothesis for the example is:

$$h(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot h^{(t)}(\boldsymbol{x})\right)$$

Decision Tress with Weighted Examples

AdaBoost is a general algorithm, but it requires that the classifiers be able to work with weighted training examples. For decision trees, this means we need to compute *weighted* conditional entropy, as well as weighted classification results at the leaves. For the j^{th} feature equalling value v, our definition of conditional entropy is:

$$H(Y|X_j = v) = \sum_{c \in \text{vals}(y)} P(Y = c|X_j = v) \log P(Y = c|X_j = v)$$

But now instead of simple counting to obtain $P(Y = c|X_j = v)$, we will use weighted counting:

$$P(Y = c|X_j = v) = \frac{\sum_{i=1}^{n} w_i^{(t)} \mathbb{1}(y_i = c, x_{ij} = v)}{\sum_{i=1}^{n} w_i^{(t)} \mathbb{1}(x_{ij} = v)}$$

We need to use a similar procedure for *every* probability involved in our entropy computations. When we reach a leaf and need to decide which label to apply, we will again use weighted counts of the training examples that fall into the partition at this leaf:

$$P(\text{leaf label is 1}) = \frac{\sum_{i \text{ in leaf}} w_i^{(t)} \mathbb{1}(y_i = 1)}{\sum_{i \text{ in leaf}} w_i^{(t)}}$$

Note that in principle, the denominator is 1 since all the weights sum to 1. However, this is only if we have been updating the weights each time we partition the data as we build the tree. We can alternatively not modify the weights during tree building (since this interacts poorly with the AdaBoost algorithm) and then just count them up in the denominator.