Randomized Decision Trees II

compiled by Alvin Wan from Professor Jitendra Malik's lecture

1 Feature Selection

Note that depth-limited trees have a finite number of combinations.

1.1 Randomized Feature Selection

Suppose X is 1000-dimensional. We can randomly select a subset of features to create a new decision tree. This is called **randomized feature selection**. If the features are nominal, such as hair color. Our questions will simply compare: black or brown? brown?

2 Boosting

Our first intuition is the "wisdom of the crowds". Our second is that we want experts for different types of samples. In other words, some trees perform better on particular samples. How do we give each tree a different weight? This note will cover only the algorithm and not the proof.

2.1 Intuition

Let us consider the boosting algorithm first proposed, trimmed.

1. Train weak learner.
2. Get weak hypothesis, $h_t : X \to \{-1, +1\}$.
3. Choose α_t.

A_t is a single decision tree with error rate ϵ_t. α_t is the weighting for a decision tree t.

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$$

First thing to notice is that ϵ_t is at least 0.5, for a binary classification problem. Any less (say, 0.45) and we can simply invert the classification for a higher accuracy (1-0.45 = 0.55). Consider the worst case scenario, where $\epsilon_t = 0.5$. Plugging in $\epsilon_t = 0.5$, we get $\alpha_t = 0$, as expected.

We pick this weighting per the error rate of a decision tree. Create a classifier h_1, compute its error ϵ_1 and weight α_1. Repeat this for the second, third etc. trees. Here is our scheme, to train an "expert". Take the samples that h_1 classified incorrectly. Train h_2 on those. Take the samples that h_2 classified incorrectly. Train h_3 on those. We can continue in this fashion to produce an "expert". This is the intuition for it, but in reality, we will instead give more weight to the samples that h_1 classified incorrectly.

2.2 Full Algorithm

Now, let us consider the original boosting algorithm in its full glory.

1. Train weak learner with distribution D_t.
2. Get weak hypothesis, $h_t : X \to \{-1, +1\}$.
3. Choose α_t.
4. Update $D_t = D_{t+1}$.

We compute a probability distribution D_t over the samples. We know that $\sum_i D_t(i) = 1$, and we can initialize D_1 to be

$$D_1(i) = \frac{1}{n}, \forall i$$

For each sample, we multiply its old weight by a factor, which effectively gives more weight to examples that were classified incorrectly. First, note that for a good classifier, α_t is high and we weight D_{t+1} less, by making our factor $e^{-\alpha t}$. If otherwise, α_t is low and our factor is $e^{\alpha t}$, to scale D_{t+1} up.

$$D_{t+1} = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

We can summarize this update as the following.

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