

Lecture 14: Boosting

Oct 22th 2019

Lecturer: Steven Wu

Scribe: Steven Wu

In this lecture, we will study an *ensemble method* called *boosting*—a general method of converting rough rules of thumb into highly accurate prediction rule. The typical rules of thumb are given by decision trees.

Decision Trees

A decision tree is a predictor function $f: \mathcal{X} \rightarrow \mathcal{Y}$, represented by a binary tree in which:

- each tree node is associated with a splitting rule $h: \mathcal{X} \rightarrow \{0, 1\}$.
- each leaf node l is associated with a fixed prediction \hat{y}_l .

Here is a fun example in Figure 1.

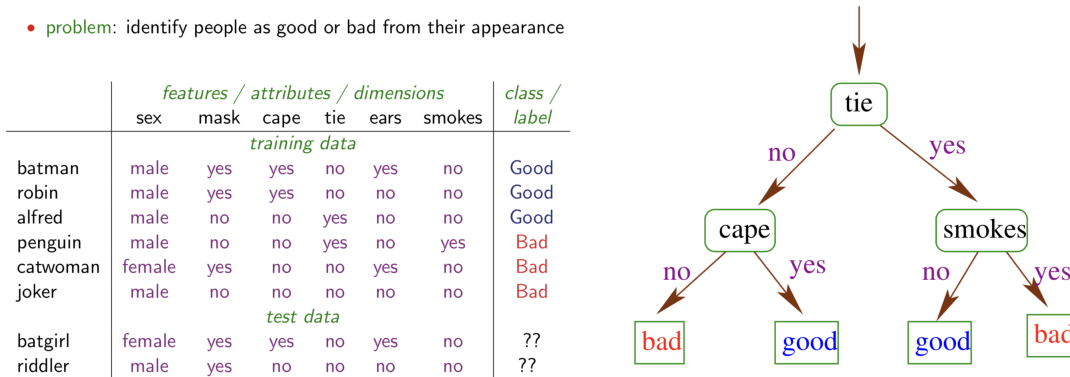


Figure 1: Fun example from Rob Schapire. Left: data; Right: decision tree predictor.

Top-down greedy training algorithm How do you build a decision tree? A typical approach is the following “top-down” greedy algorithm:

- Initialize a tree with a single leaf node containing all training data.
- While “stopping criterion” is not met:

- Pick the leaf l and rule h that maximally reduces “uncertainty” measure u :

$$\sum_l w(l)u(S_l)$$

where $w(l)$ denotes the fraction of examples reaching the leaf l and S_l denote the set of examples reaching leaf l .

- Split data in l using h , and grow tree accordingly
- Label \hat{y}_l of each leaf l is the majority label (for classification) or average label value (for regression) among the data contained in the leaf.

For this algorithm, we need to define the uncertainty measure u as well as stopping criterion.

Notions of uncertainty (for binary classification). Suppose in a set of examples S , the fraction of positive examples is p . Then we can define three uncertainty measures.

1. Classification error (for the majority prediction rule):

$$u(S) = \min\{p, 1 - p\}$$

2. Gini index:

$$u(S) = 2p(1 - p)$$

3. Entropy:

$$u(S) = p \log(1/p) + (1 - p) \log(1/(1 - p))$$

Note that both Gini index and rescaled entropy are upper bounds on the classification error. See Figure 2 for an illustration.

Notion of uncertainty for regression.

$$u(S) = \frac{1}{|S|} \sum_{(x,y) \in S} (y - \bar{y}_S)^2 \leftarrow \text{Average squared difference from average label}$$

$$\text{where } \bar{y}_S = \frac{1}{|S|} \sum_{(x,y) \in S} y \leftarrow \text{Average label}$$

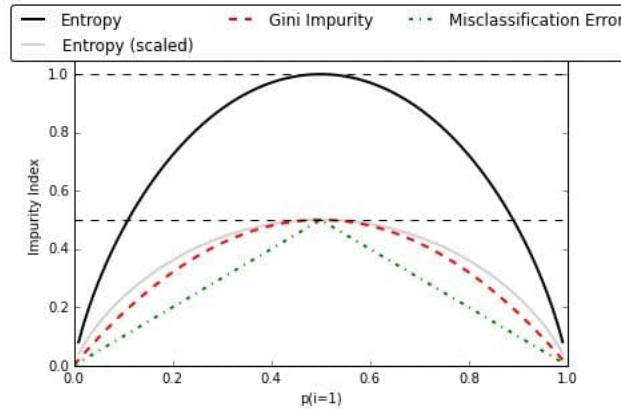


Figure 2: Three standard uncertainty or impurity functions. Image source.

Stopping criteria. Two common choices:

- Stop when the tree reaches a pre-specified size (given by depth or number of nodes). Tree size is a hyperparameter we need to tune.
- Stop when every leaf is pure, that is the examples in each leaf belong to the same class. This will often lead to very large trees, which can result in a risk of overfitting. To mitigate over-fitting, we can perform *tree pruning*: first, split training data S into two parts S_1 and S_2 , and then
 - Use S_1 to grow the tree until all leaves are pure.
 - Use S_2 to choose a good pruning of the tree: Replace any tree node by a leaf node if it improves the error on S_2 until no more such improvements possible.

Boosting

Boosting combines weak predictor that has say 51% accuracy to form a highly accurate predictor that say 99% accuracy. It is rooted in learning theory, and works very well in practice (especially in combination with trees). Here is the general approach:

- take a method for deriving rough rules of thumb
- apply the method to subset of examples
- obtain rule of thumb
- apply to 2nd subset of examples
- obtain 2nd rule of thumb

- repeat...

Two questions arise:

1. *How to choose examples on each round?* We should concentrate on “hardest” examples, that is those misclassified by previous weak rules of thumb.
2. *How to combine rules of thumb into single prediction rule?* Take weighted majority vote of rules of thumb.

AdaBoost is an elegant boosting method that implements these ideas. The algorithm takes as input a training dataset: $(x_1, y_1), \dots, (x_n, y_n) \in \mathcal{X} \times \{\pm 1\}$:

- Initialize D_1 as the uniform distribution over the examples.
- For $t = 1, \dots, T$:
 - Train weak classifier (“rule of thumb”) h_t on D_t
 - Let $\epsilon_t = \sum_i D_t(i) \mathbf{1}[h_t(x_i) \neq y_i]$ and choose parameter α_t

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

- compute new distribution D_{t+1} : for each example i , the weight is

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

(\propto means “proportional to”, which hides the normalization step.)

- Output final classifier: $\hat{f}: x \rightarrow \text{sign}(\sum_t \alpha_t h_t(x))$

Let’s work through an example (due to Rob Schapire). We will use *decision stump* of the form $\mathbf{1}[x_j \geq \theta]$ as our weak predictors. (Here I am abusing notation to write x_j as the j -th coordinate of x .) Suppose we are given training examples shown in Figure 3. Then Figure 4 shows a visualization of how AdaBoost updates over rounds, which leads to a final predictor with zero training error (in Figure 5).

Under some “weak learning” assumption, one can show that the training error goes to zero exponentially fast.

Theorem 0.1. *Suppose the weak learning assumption holds for all t : each h_t is better than random guessing: for some $\gamma > 0$,*

$$\epsilon_t \leq 1/2 - \gamma$$

Then the training error

$$\hat{\mathcal{R}}_{01}(\hat{f}) \leq \exp(-2\gamma^2 T).$$

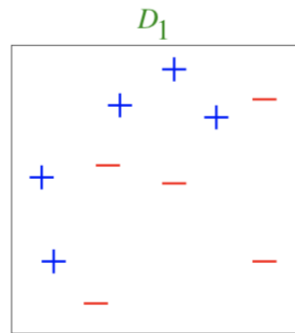
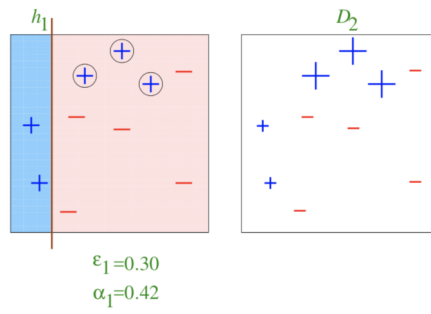
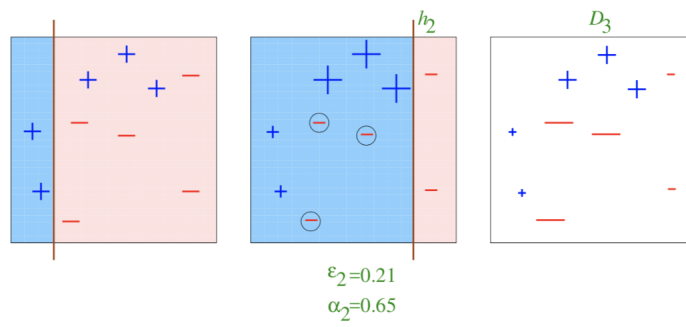


Figure 3: Training data.

Round 1



Round 2



Round 3

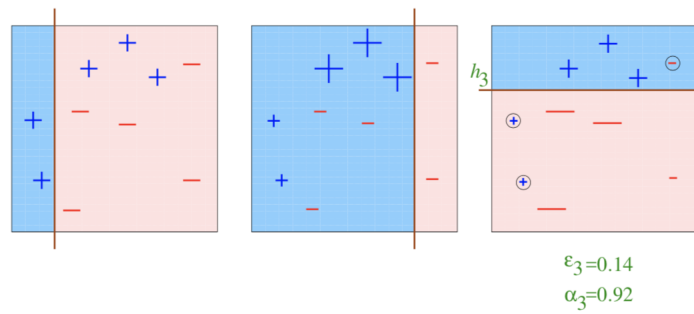


Figure 4: Adaboost re-weighting over rounds.

Final Classifier

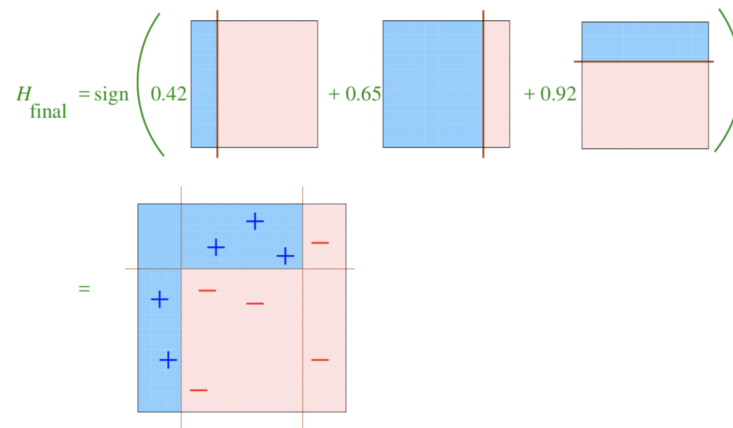


Figure 5: Final predictor with perfect accuracy.