Lec 20: Tree and Boosting

Ailin Zhang

Agenda

- Basic idea
- Weak learner example: classification tree
- Decision Tree and Regression Tree
- L2 boosting
- AdaBoost
- Gradient Boosting
- Extreme Gradient Boosting (XGB)
- Connection to other models

Basic Idea

Boosting algorithms generally attempt to best fit the data by making use of **weak-learners**, where each of these is found via sequential pursuit, to minimize a given loss function

Basic Idea

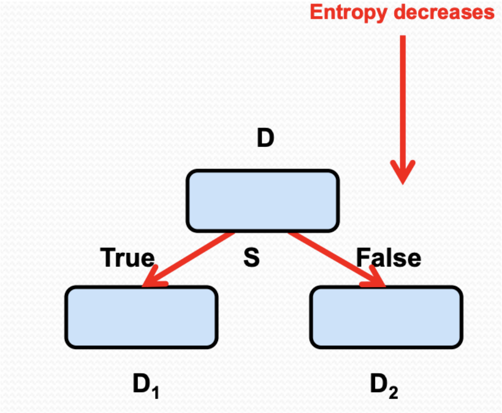
Boosting algorithms generally attempt to best fit the data by making use of **weak-learners**, where each of these is found via sequential pursuit, to minimize a given loss function

We look for a function of the type $f(x) = \sum_{m=1}^{M} \beta_m h_m(x)$ that help us minimize the loss, where the h_m terms are weak-learners (they can be tree stumps, for example), found in a sequential order.

Boosting can also be treated as

- Ensemble machine
- Committee machine

Decision Tree



- Gini-index: $\hat{p}(1-\hat{p})$
- Overall Gini-index: $\sum n_i \hat{p}_i(1-\hat{p}_i)$

Greedy Algorithm

In each iteration:

- Choose a region:
  - Choose a variable to partition on:
    - Choose a cut point to maximize the reduction loss

Perform exhaustive search to determine optimal partitioning.

Regression Tree

Sketch on board

L2 Boosting

Goal: Find $f(x) = \sum_{m=1}^{M} \beta_m h_m(x)$ that help us minimize squared-loss, where the h_m terms are weak-learners.

Algorithm:

Let

$$L(y_i, f(x_i)) = (y_i - f(x_i))^2.$$

Then the objective is

$$\min \sum_{i=1}^{n} L(y_i, f(x_i))$$

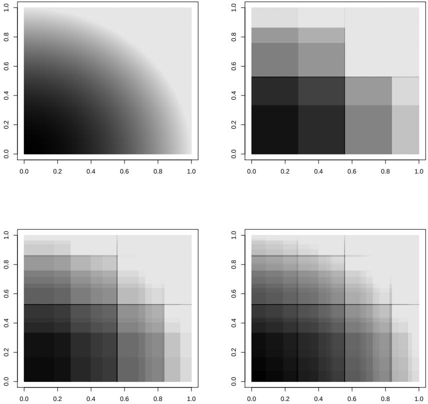
L2 Boosting: Algorithm

- *Step 1*: Given training data $\{(y_i, x_i); i = 1, \cdots, n\}$, we first fit the initial base learner $\hat{f}_0(x)$.
- *Step 2*: Compute residuals $u_i = y_i - f_m(x_i)$. Then we fit another base learner for the residuals, which gives us $g(x_i)$. Then we update

$$\hat{f}_{m+1}(x_i) = \hat{f}_m(x_i) + g(x_i).$$

- *Step 3*: $m \leftarrow m + 1$; repeat Step 2.

L2 Boosting



(top-left) function $f(\cdot)$, (top-right) weak-learners $M = 10$, (bottom-left) weak-learners $M = 40$, (bottom-right) weak-learners $M = 200$.

Adaboost (Adaptive Boosting)

AdaBoost algorithm is analogous to $L2$ boosting,

except that here, we are interested in a classification-type problem;

$$f(x) = \sum_{m=1}^{M} \beta_m h_m(x) \quad \rightarrow \quad \text{classifier} \quad \rightarrow \quad \{+1, -1\}$$

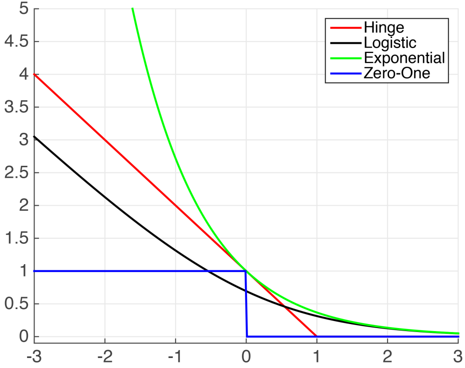
$$\hat{y} = sign(f(x))$$

we use exponential loss as opposed to squared-loss; and the βs are found in a different fashion.

Consider any stage of iteration of the standard boosting procedure, when exponential loss is used. $(\exp(-y_i f(x_i)))$ The goal at this stage can be written as:

$$min\{\sum_{i=1}^{M} \exp(-y_i \hat{f}(x_i)) e^{-y_i \beta h(x_i)}\}$$

Exponential Loss



Loss functions for classification. The horizontal axis is $m_i = y_i X_i^\top \beta$. The vertical axis is $L(y_i, X_i^\top \beta)$.

Loss function summary

Possible choices for the loss function $L(y_i, X_i^\top \beta)$ for classification.

$$\text{Logistic loss} = \log\left(1 + \exp(-y_i X_i^\top \beta)\right),$$
$$\text{Exponential loss} = \exp\left(-y_i X_i^\top \beta\right),$$
$$\text{Hinge loss} = \max\left(0, 1 - y_i X_i^\top \beta\right),$$
$$\text{Zero-one loss} = 1\left(y_i X_i^\top \beta > 0\right)$$

Both the exponential and hinge losses can be considered approximations to the logistic loss.

- The logistic loss is used by logistic regression.
- The exponential loss is used by adaboost.
- The hinge loss is used by support vector machine.
- The zero-one loss is to count the number of mistakes. It is not differentiable and is not used for training.

Adaboost

Denote $w_i = exp(-y_i \hat{f}(x_i))$. w_i tells us how well example i is classified. If w_i is large, the example is not classified well. We can re-write this as:

$$\sum_{y_i = h(x_i)} w_i e^{-\beta} + \sum_{y_i \neq h(x_i)} w_i e^{\beta}$$

Let $a = \sum_{y_i = h(x_i)} w_i$ and $b = \sum_{y_i \neq h(x_i)} w_i$. And $a + b = \sum_i^n w_i = s$.

The problem then becomes:

$$(s - b)e^{-\beta} + be^{\beta} = se^{-\beta} + b(e^{\beta} - e^{-\beta})$$

This is minimized when we minimize b, so essentially we want at each stage of the boosting procedure to find a weak-classifier h such that $h = argmin_h \{\sum_{y_i \neq h(x_i)} w_i\} = argmin_h \{\sum_{y_i \neq h(x_i)} w_i / s\}$.

Adaboost

Let

$$\epsilon = b/s = \frac{\sum_{y_i \neq h(x_i)} w_i}{\sum_i w_i}$$

This is equivalent to re-weighting the data using weights w_i/s at each stage, and then finding a classifier h that best classifies the re-weighted data.

Here, we see that the re-weighted data will have larger weights on examples that have been classified poorly - so the re-weighted data are analogous to the residuals that result from each iteration stage in $L2$ boosting.

R code for adaboost

```
myAdaboost <- function(X_train, Y_train, X_test, Y_test,num_iterations = 200)
{
n <- dim(X_train)[1]
p <- dim(X_train)[2]
threshold <- 0.8
X_train1 <- 2 * (X_train > threshold) - 1
Y_train  <- 2 * Y_train - 1
X_test1 <- 2 * (X_test > threshold) - 1
Y_test <- 2 * Y_test-1
beta <- matrix(rep(0,p), nrow = p)
w <- matrix(rep(1/n, n), nrow = n)
weak_results <- Y_train * X_train1 > 0
acc_train <- rep(0, num_iterations)
acc_test <- rep(0, num_iterations)

for(it in 1:num_iterations)
{
  w <- w / sum(w)
  weighted_weak_results <- w[,1] * weak_results
  weighted_accuracy <- colSums(weighted_weak_results)
  e <- 1 - weighted_accuracy
  j <- which.min(e)
  dbeta <-log((1-e[j])/e[j])/2
  beta[j] <- beta[j] + dbeta
  w <-  w*exp(-y*x[, j]*dbeta)
  acc_train[it] <- mean(sign(X_train1 %*% beta) == Y_train)
  acc_test[it] <- mean(sign(X_test1 %*% beta) == Y_test)
}
  output <- list(beta = beta, acc_train = acc_train, acc_test = acc_test)
output
}
```