Training Objective

To understand other parameters, one need to have a basic understanding of the model behind.

Suppose we have K trees, the model is

$$\sum_{k=1}^{K} f_k$$

where each f_k is the prediction from a decision tree. The model is a collection of decision trees.

Training Objective

Having all the decision trees, we make prediction by

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$$

where x_i is the feature vector for the i-th data point.

Similarly, the prediction at the t-th step can be defined as

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i)$$

Training Objective

To train the model, we need to optimize a loss function.

Typically, we use

- Rooted Mean Squared Error for regression
  - $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
- LogLoss for binary classification
  - $L = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$
- mlogloss for multi-classification
  - $L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$

Training Objective

Regularization is another important part of the model. A good regularization term controls the complexity of the model which prevents overfitting.

Define

$$\Omega = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

where T is the number of leaves, and w_j^2 is the score on the j-th leaf.

Training Objective

Put loss function and regularization together, we have the objective of the model:

$$Obj = L + \Omega$$

where loss function controls the predictive power, and regularization controls the simplicity.

Training Objective

In XGBoost, we use gradient descent to optimize the objective.

Given an objective $Obj(y, \hat{y})$ to optimize, gradient descent is an iterative technique which calculate

$$\partial_{\hat{y}} Obj(y, \hat{y})$$

at each iteration. Then we improve $\hat{y}$ along the direction of the gradient to minimize the objective.

Training Objective

Recall the definition of objective $Obj = L + \Omega$. For a iterative algorithm we can re-define the objective function as

$$Obj^{(t)} = \sum_{i=1}^{N} L(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) = \sum_{i=1}^{N} L(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \sum_{i=1}^{t} \Omega(f_i)$$

To optimize it by gradient descent, we need to calculate the gradient. The performance can also be improved by considering both the first and the second order gradient.

$$\partial_{\hat{y}_i^{(t)}} Obj^{(t)}$$

$$\partial^2_{\hat{y}_i^{(t)}} Obj^{(t)}$$

Training Objective

Since we don't have derivative for every objective function, we calculate the second order taylor approximation of it

$$Obj^{(t)} \simeq \sum_{i=1}^{N} [L(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i {f_t}^2(x_i)] + \sum_{i=1}^{t} \Omega(f_i)$$

where

- $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$
- $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

Training Objective

Remove the constant terms, we get

$$Obj^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

This is the objective at the t-th step. Our goal is to find a f_t to optimize it.

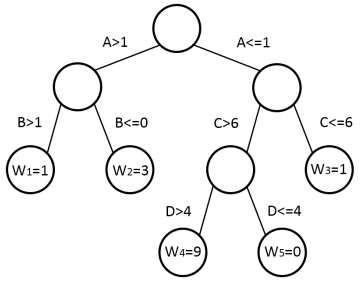
Tree Building Algorithm

The tree structures in XGBoost leads to the core problem:

how can we find a tree that improves the prediction along the gradient?

Tree Building Algorithm

Every decision tree looks like this



Each data point flows to one of the leaves following the direction on each node.

Tree Building Algorithm

The core concepts are:

- Internal Nodes
  - Each internal node split the flow of data points by one of the features.
  - The condition on the edge specifies what data can flow through.
- Leaves
  - Data points reach to a leaf will be assigned a weight.
  - The weight is the prediction.

Tree Building Algorithm

Two key questions for building a decision tree are

1. How to find a good structure?
2. How to assign prediction score?

We want to solve these two problems with the idea of gradient descent.

Tree Building Algorithm

Let us assume that we already have the solution to question 1.

We can mathematically define a tree as

$$f_t(x) = w_{q(x)}$$

where $q(x)$ is a "directing" function which assign every data point to the $q(x)$-th leaf.

This definition describes the prediction process on a tree as

- Assign the data point x to a leaf by q
- Assign the corresponding score $w_{q(x)}$ on the $q(x)$-th leaf to the data point.

Tree Building Algorithm

Define the index set

$$I_j = \{i|q(x_i) = j\}$$

This set contains the indices of data points that are assigned to the j-th leaf.

Tree Building Algorithm

Then we rewrite the objective as

$$Obj^{(t)} = \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2} h_i {f_t}^2(x_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T$$

Since all the data points on the same leaf share the same prediction, this form sums the prediction by leaves.

Tree Building Algorithm

It is a quadratic problem of w_j, so it is easy to find the best w_j to optimize Obj.

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

The corresponding value of Obj is

$$Obj^{(t)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T$$

Tree Building Algorithm

The leaf score

$$w_j = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

relates to

- The first and second order of the loss function g and h
- The regularization parameter λ

Tree Building Algorithm

Now we come back to the first question: How to find a good structure?

We can further split it into two sub-questions:

1. How to choose the feature to split?
2. When to stop the split?

Tree Building Algorithm

In each split, we want to greedily find the best splitting point that can optimize the objective.

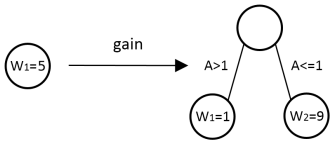
For each feature

1. Sort the numbers
2. Scan the best splitting point.
3. Choose the best feature.

Tree Building Algorithm

Now we give a definition to "the best split" by the objective.

Everytime we do a split, we are changing a leaf into a internal node.



Tree Building Algorithm

Let

- I be the set of indices of data points assigned to this node
- I_L and I_R be the sets of indices of data points assigned to two new leaves.

Recall the best value of objective on the j-th leaf is

$$Obj^{(t)} = -\frac{1}{2}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma$$

Tree Building Algorithm

The gain of the split is

$$gain = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

Tree Building Algorithm

To build a tree, we find the best splitting point recursively until we reach to the maximum depth.

Then we prune out the nodes with a negative gain in a bottom-up order.

Tree Building Algorithm

XGBoost can handle missing values in the data.

For each node, we guide all the data points with a missing value

- to the left subnode, and calculate the maximum gain
- to the right subnode, and calculate the maximum gain
- Choose the direction with a larger gain

Finally every node has a "default direction" for missing values.

Tree Building Algorithm

To sum up, the outline of the algorithm is

- Iterate for `nround` times
  - Grow the tree to the maximun depth
    - Find the best splitting point
    - Assign weight to the two new leaves
  - Prune the tree to delete nodes with negative gain