UNIVERSITY OF ARKANSAS

Gradient Boosting Tree

Jiahui Chen
Department of Mathematical Sciences
University of Arkansas

Introduction

- Gradient Boosting = Gradient descent + Boosting
- Boosting: at each stage, introduce a weak learner to compensate the shortcomings of existing weak learners
- The first successful boosting method: Adaboost (Freund et. al. , ICML, 1997)
- Utilize gradient descent in Adaboost to deal with a special loss function (Breiman et. al., Ann. Stat., 1998)
- Generalize Adaboost to Gradient Boosting for handling various of loss functions (Friedman, Ann. Stat., 2001)

Trees in Gradient Boosting (Regression version)

- Given a training data: $(X_1, y_1), (X_2, y_2), \dots, (X_n, y_n)$

We build the gradient boosting trees to fit that data as follows

- Build a decision Tree 1 to fit $(X_1, y_1), (X_2, y_2), \dots, (X_n, y_n)$.

The predicted labels are $\hat{y}_1^{(1)}, \hat{y}_2^{(1)}, \dots, \hat{y}_n^{(1)}$

The errors (residues) are $r_i^{(1)} = y_i - \hat{y}_i^{(1)}$

If Tree 1 does not fit $\{(X_i, y_i)\}_{i=1}^{n}$ perfectly, then $r_i^{(1)} \neq 0$ for some i.

Trees in Gradient Boosting (Regression version)

- Build a decision Tree 2 to compensate the shortcoming of the existing model (Tree 1). In other words, Tree 2 is constructed to fit the following data $\left(X_1, r_1^{(1)}\right), \left(X_2, r_2^{(1)}\right), \ldots, \left(X_n, r_n^{(1)}\right)$.

The predicted labels of Tree 2 are $\hat{y}_1^{(2)}, \hat{y}_2^{(2)}, \ldots, \hat{y}_n^{(2)}$

The errors (residues) are

$$r_i^{(2)} = r_i^1 - \hat{y}_i^{(2)} = y_i - \hat{y}_i^{(1)} - \hat{y}_i^{(2)}$$

The predicted labels for $\{(X_i, y_i)\}_{i=1}^n$ by combining Tree 1 and Tree 2 are $\hat{y}_i^{(1)} + \hat{y}_i^{(2)}$

Trees in Gradient Boosting (Regression version)

- If $\left|r_i^{(2)}\right| \gg 0$, we may build Tree 3 to fit $\left\{\left(X_i, r_i^{(2)}\right)\right\}$.
- The predicted values for the data $\{(X_i, y_i)\}_{i=1}^{n}$ based K consecutive decision trees are

$$\hat{y}_i = \sum_{j=1}^{K} \hat{y}_i^{(j)}$$

Residues and Gradient Descent

- So far, we have just formulated the Boosting Trees.
- How can we make use of the gradient descent?
- Consider a general loss function

$$L = \sum_i l_i(y_i, \hat{y}_i)$$

- If we consider a square loss: $l_i = \frac{(y_i - \hat{y}_i)^2}{2}$
- The total loss function L can be minimized along the following gradient direction

$$-\frac{\partial L}{\partial \hat{y}_i} = -\frac{\partial}{\partial \hat{y}_i} \sum_i l_i(y_i, \hat{y}_i) = y_i - \hat{y}_i = \text{residues}$$

Loss Functions

- In the Boosting Trees, we update the decision trees using the residues. When we extend to the Gradient Boosting Tree, we update the model based on the gradient descent.
- Gradient descent depends on the loss functions
- Besides the square loss, we can employ the following loss functions to handle the **outliers**
- Absolute loss: $L(y, \hat{y}) = |y - \hat{y}|$
- Huber loss:

$$L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2, \text{if } |y - \hat{y}| \leq \delta \\ \delta\left(|y - \hat{y}| - \frac{\delta}{2}\right), \text{if} |y - \hat{y}| > \delta \end{cases}$$

Loss Functions

Outliers

y	0.5	1.2	2	5
$\hat{y}$	0.6	1.4	1.5	1.7
Square loss	0.005	0.02	0.125	5.445
Abs. loss	0.1	0.2	0.5	3.3
Huber loss ($\delta = 0.5$)	0.005	0.002	0.125	1.525

- In the square loss, the error of the outlier target is significantly bigger than the rest. If we try hard to incorporate this outlier to the model, it will degrade the overall performance.

Loss Functions

- Regression with absolute loss

$$r_i = -\frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i} = -\frac{\partial |y_i - \hat{y}_i|}{\partial \hat{y}_i} = \text{sign}(y_i - \hat{y})$$

- Regression with Huber loss

$$r_i = -\frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i} = \begin{cases} y_i - \hat{y}_i, \text{if } |y - \hat{y}| \le \delta \\ \delta \text{sign}(y_i - \hat{y}), \quad \text{if } |y - \hat{y}| > \delta \end{cases}$$

Gradient Boosting Trees Algorithm

- Initialize model: $\hat{y}_i^{(0)} = p^{(0)}(X_i) = \frac{1}{n}\sum_i^n y_i$
- Build sequentially M decision trees

For $m = 1$ to M:

1. Calculate gradient

$$r_i^{(m)} = -\frac{\partial L\left(y_i, p^{(m-1)}(X_i)\right)}{\partial \hat{y}_i}$$

2. Construct a decision tree $T^{(m)}$ to fit $\left\{\left(X_i, r_i^{(m)}\right)\right\}$

Let $h^{(m)}(X)$ is the predictor of $T^{(m)}$

Gradient Boosting Trees Algorithm (cont’d)

3. Choose learning rate $\alpha^{(m)}$ such that

$$\alpha^{(m)} = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, p^{(m-1)}(X_i) + \alpha h^{(m)}(X_i))$$

4. Update model

$$p^{(m)}(X) = p^{(m-1)}(X) + \eta \alpha^{(m)} h^{(m)}(X)$$

In which η is a shrinkage, predefined parameters $(0 < \eta \leq 1)$

Important Parameters of GBT in Sklearn package

- Load GBT regressor module:

```
Import sklearn.ensemble.GradientBoostingRegressor
```

- `n_estimators` (number of trees)
- `learning_rate` (shrinkage in our algorithm)
- `max_features`
- `subsample`
- `max_depth`
- `min_samples_split`

Gradient Boosting for Classification

- Probability in the leaf node

Assume our data contains only dog and cat. Suppose we have two following leaf nodes

Dog: 1
Cat: 2

Leaf node 1

Dog: 10
Cat: 20

Leaf node 2

- Basically, we have two ways to determine the class probability

1. Standard probability: $p(Cat) = \frac{n_{cat}}{n_{cat}+n_{dog}}$
2. Softmax function: $p(Cat) = \frac{e^{n_{cat}}}{e^{n_{cat}}+e^{n_{dog}}}$

Gradient Boosting for Classification

- In practice, the softmax function is more commonly used since it takes into account the size of the classes
- Under the standard probability

In both node 1 and node 2:

$$p(Cat) \approx 0.67, p(Dog) \approx 0.33$$

- Under the softmax function

In node 1: $p(Cat) \approx 0.73, p(Dog) \approx 0.27$

In node 2: $p(Cat) \approx 1.0, p(Dog) \approx 0.0$

Gradient Boosting for Classification: Loss Function

- Assume the data $\{(X_i, y_i)\}_{i=1}^{n}$ has K classes denoted by $L_1, L_2, \dots, L_K$. Under the one-hot-encoder fashion, we obtain a data with a new label representation $\{(X_i, z_i)\}_{i=1}^{n}$
- Loss functions
  - Kullback-Leibler divergence

$$L(z, \hat{z}) = -\sum_i z_i \log \frac{\hat{z}_i}{z_i}$$

Gradient Boosting for Classification: Loss Function

- Initialize models:

$$\hat{z}_i = (p_1^{(0)}(X_i), p_2^{(0)}(X_i), \dots, p_K^{(0)}(X_i)) = (0.5, 0.5, \dots, 0.5)$$

- Build sequentially M decision trees

For $m = 1$ to M:

1. Calculate K negative gradient

$$r_{i,k}^{(m)} = -\frac{\partial L\left(z_{i,k}, p_k^{(m-1)}(X_i)\right)}{\partial \hat{z}_{i,k}}, k = 1, \dots, K$$

2. Construct K regression tree $T_k^{(m)}$ to fit $\left\{\left(X_i, r_{i,k}^{(m)}\right)\right\}$

Gradient Boosting for Classification: Algorithm (Cont'd)

3. Choose learning rate $\alpha^{(m)}$ such that

$$\alpha_k^{(m)} = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} L(z_{i,k}, p_k^{(m-1)}(X_i) + \alpha h_k^{(m)}(X_i))$$

$h_k^{(m)}$ is the predictor of the decision tree $T_k^{(m)}$

4. Update model

$$p_k^{(m)}(X) = p_k^{(m-1)}(X) + \eta \alpha_k^{(m)} h_k^{(m)}(X)$$

In which η is a shrinkage, predefined parameters $(0 < \eta \leq 1)$