Gradient Boosted Decision Tree

Yafei Zhang

kimmyzhang@tencent.com

August 1, 2016

Outline

Decision Tree

1. J. Quinlan, 1986, ID3
   - Classification

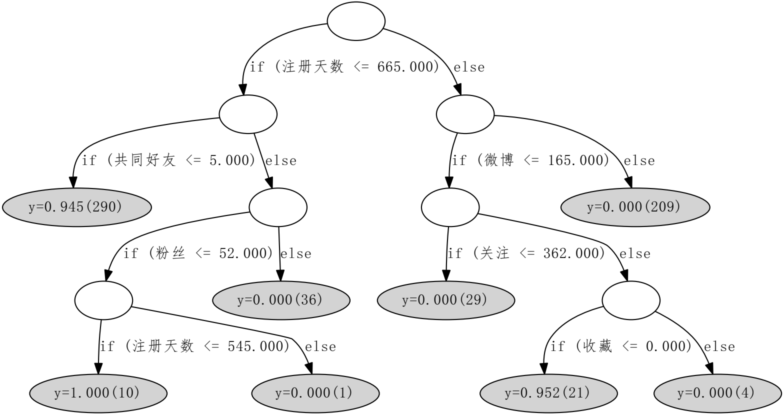
Decision Tree

1. J. Quinlan, 1986, ID3
   - Classification
2. J. Quinlan, 1993, C4.5
   - Classification

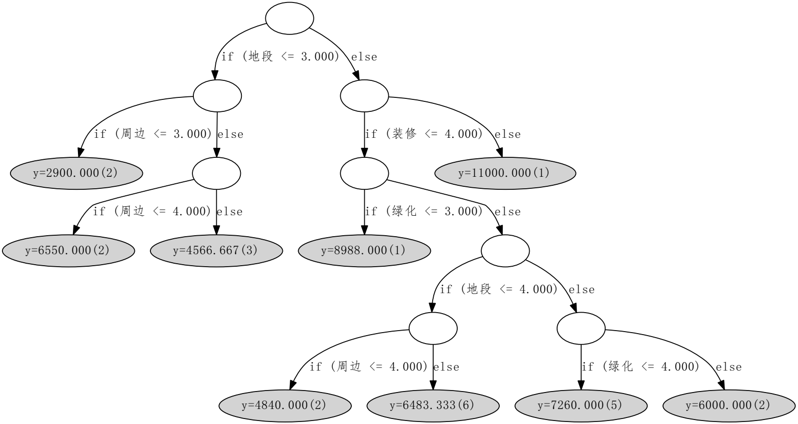
Decision Tree

1. J. Quinlan, 1986, ID3
   - Classification
2. J. Quinlan, 1993, C4.5
   - Classification
3. L. Breiman and J. Friedman, 1984, CART
   - Classification
   - Regression

A Classification Demo: Weibo Spam Users



A Regression Demo: Real Estate Price



Train a Decision Tree

Split training set at "the best value" of "the best feature"

- Information gain (ratio)
- Gini index
- Mean square error

Train a Decision Tree

Split training set at "the best value" of "the best feature"

- Information gain (ratio)
- Gini index
- Mean square error

Decide leaf value

- # of Positive/# of Total
- Mean

Train a Decision Tree

Split training set at "the best value" of "the best feature"

- Information gain (ratio)
- Gini index
- Mean square error

Decide leaf value

- # of Positive/# of Total
- Mean

Split its children recursively until termination criteria are met

- # of leaves
- Tree depth
- # of instances in current node

Train a Decision Tree

Split training set at "the best value" of "the best feature"

- Information gain (ratio)
- Gini index
- Mean square error

Decide leaf value

- # of Positive/# of Total
- Mean

Split its children recursively until termination criteria are met

- # of leaves
- Tree depth
- # of instances in current node

Post-Prune

Advantages and Disadvantages of Decision Tree

Advantages

- Invariant to scale of feature, need less data preprocessing
  - But it never means we don't need to clean and transform features.
- Naturally support categorical feature
- Easy to understand, interpret and implement

Advantages and Disadvantages of Decision Tree

Advantages

- Invariant to scale of feature, need less data preprocessing
  - But it never means we don't need to clean and transform features.
- Naturally support categorical feature
- Easy to understand, interpret and implement

Disadvantages

- Overfitting
- Limited fitting capability

Model Aggregation

http://en.wikipedia.org/wiki/Ensemble_learning
Model Ensemble(Aggregation, Blending, Combination)

Model Aggregation

http://en.wikipedia.org/wiki/Ensemble_learning

Model Ensemble(Aggregation, Blending, Combination)

- Bagging(Bootstrap Aggregation)
  - Random Forest
  - ...

Model Aggregation

http://en.wikipedia.org/wiki/Ensemble_learning

Model Ensemble(Aggregation, Blending, Combination)

- Bagging(Bootstrap Aggregation)
  - Random Forest
  - ...
- Boosting
  - AdaBoost
  - Gradient Boosting
    - GBDT
  - ...

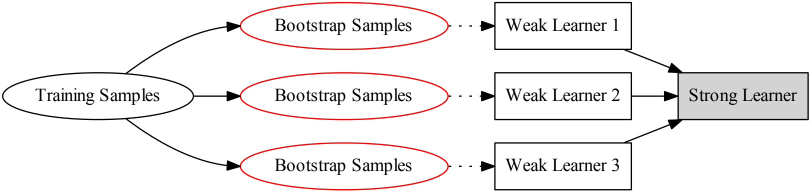
Model Aggregation

http://en.wikipedia.org/wiki/Ensemble_learning

Model Ensemble(Aggregation, Blending, Combination)

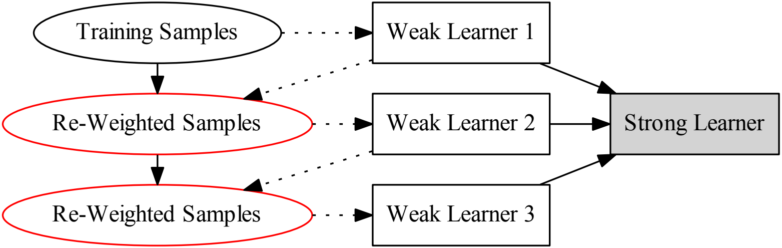
- Bagging(Bootstrap Aggregation)
    - Random Forest
    - ...
- Boosting
    - AdaBoost
    - Gradient Boosting
        - GBDT
    - ...
- ...

Bagging



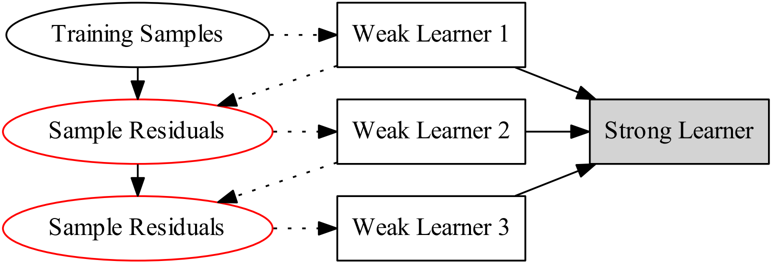
Weak learners are independent.
Bagging can be easily parallelized.

Adaboost



Guarantee: sum of unnormalized weights is monotonously decreasing.
Sum of unnormalized weights measures the error.

Gradient Boosting



Guarantee: sum of residuals is monotonously decreasing.
Residuals are highly related to loss.

Tree Ensemble(Additive Tree)

A decision tree h

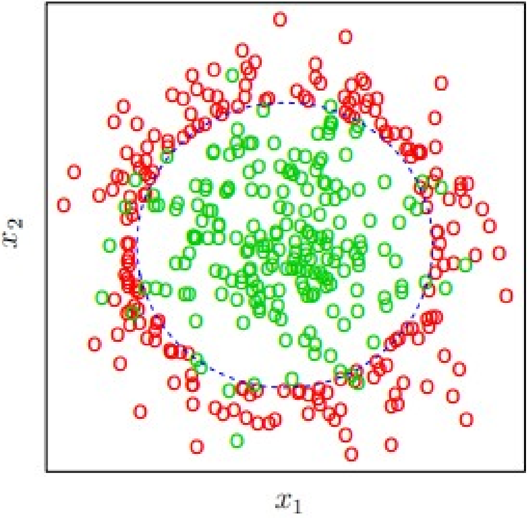
$$h(x; \{b_j, R_j\}_1^J) = \sum_{j=1}^{J} b_j \mathbf{1}[x \in R_j] \tag{1}$$

A tree ensemble F

$$F(x; w) = \sum_{k=0}^{K} \alpha_k h_k(x; \{b_j, R_j\}_1^{J_k}) \tag{2}$$

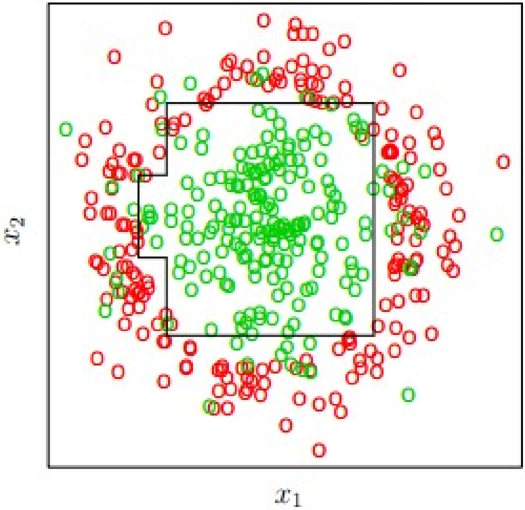
J: # of leaves.
$\{b_j\}_1^J$: values given by leaves.
$\{R_j\}_1^J$: disjoint regions that cover the domain of x.
K: # of trees.
$\{\alpha_k\}_1^K$: weights of each decision tree.

A 2D Demo: Raw Data



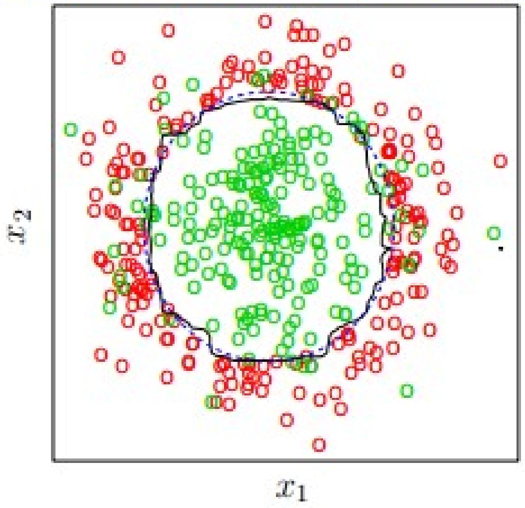
Picture from Internet

A 2D Demo: Decision Boundary of One Decision Tree



Picture from Internet

A 2D Demo: Decision Boundary of the Tree Ensemble



Picture from Internet

Outline

Notations

A training set $\mathcal{D} = \{(x_i, y_i)\}_1^N$. A loss function L. The model F.

Notations

A training set $\mathcal{D} = \{(x_i, y_i)\}_1^N$. A loss function L. The model F.

F is an additive model

$$F(x; w) = \sum_{k=0}^{K} \alpha_k h_k(x; w_k) = \sum_{k=0}^{K} f_k(x; w_k) \tag{3}$$

Define:

$$F_k = \sum_{i=0}^{k} f_i \tag{4}$$

$\{h_k(x; w_k)\}_1^K$, $\{\alpha_k\}_1^K$, $\{w_k\}_1^K$: weak learners and their weights, parameters.

Goal

Overall Loss Function

$$\mathcal{L} = \underbrace{\sum_{i=1}^{N} L(y_i, F(x_i; w))}_{Training\ \ loss} + \underbrace{\sum_{k=1}^{K} \Omega(f_k)}_{Regularization} \tag{5}$$

Goal

Overall Loss Function

$$\mathcal{L} = \underbrace{\sum_{i=1}^{N} L(y_i, F(x_i; w))}_{Training\ \ loss} + \underbrace{\sum_{k=1}^{K} \Omega(f_k)}_{Regularization} \tag{5}$$

Goal

$$F^* = \arg\min_F \mathcal{L} \tag{6}$$

Goal

Overall Loss Function

$$\mathcal{L} = \underbrace{\sum_{i=1}^{N} L(y_i, F(x_i; w))}_{Training\ \ loss} + \underbrace{\sum_{k=1}^{K} \Omega(f_k)}_{Regularization} \tag{5}$$

Goal

$$F^* = \arg\min_F \mathcal{L} \tag{6}$$

This is a NP hard problem.

Learn Greedily

Iteration 0

Choose a f_0, usually a constant.

Learn Greedily

Iteration 0

Choose a f_0, usually a constant.

Iteration k

$$f_k = \arg\min_{f_k} \mathcal{L}(f_k) \tag{7}$$

$$= \arg\min_{f_k} \sum_{i=1}^{N} L(y_i, F_{k-1}(x_i; w) + f_k(x_i)) + \Omega(f_k) \tag{8}$$

Learn Greedily

Iteration 0

Choose a f_0, usually a constant.

Iteration k

$$f_k = \arg\min_{f_k} \mathcal{L}(f_k) \tag{7}$$

$$= \arg\min_{f_k} \sum_{i=1}^{N} L(y_i, F_{k-1}(x_i; w) + f_k(x_i)) + \Omega(f_k) \tag{8}$$

Finally

$$F^* = \sum_{k=1}^{K} f_k \tag{9}$$

Gradient Boosting Machine

$\Omega(f_k) = 0$ in [Friedman(1999)].

1 choose an initial f_0, let $F_0 = f_0$

2 for $k = 1, 2, \dots, K$

  2.1 $\tilde{y}_i = -\frac{\partial L(y_i, F_{k-1}(x_i))}{\partial F_{k-1}(x_i)}$, $i = 1, 2, \dots, N$

  2.2 $w^* = \arg\min_w \sum_{i=1}^{N} [\tilde{y}_i - h_k(x_i; w)]^2$

  2.3 $\rho^* = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{k-1}(x_i) + \rho h_k(x_i; w^*))$

  2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$

3 output F_K

Gradient Boosting Machine

$\Omega(f_k) = 0$ in [Friedman(1999)].

1 choose an initial f_0, let $F_0 = f_0$

2 for $k = 1, 2, \ldots, K$

2.1 $\tilde{y}_i = -\frac{\partial L(y_i, F_{k-1}(x_i))}{\partial F_{k-1}(x_i)}$, $i = 1, 2, \ldots, N$

$\{\tilde{y}_i\}_1^N$: pseudo responses, a measurement of residuals, namely $\{y_i - F_{k-1}(x_i)\}_1^N$

2.2 $w^* = \arg\min_w \sum_{i=1}^N [\tilde{y}_i - h_k(x_i; w)]^2$

2.3 $\rho^* = \arg\min_\rho \sum_{i=1}^N L(y_i, F_{k-1}(x_i) + \rho h_k(x_i; w^*))$

2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$

3 output F_K

Gradient Boosting Machine

$\Omega(f_k) = 0$ in [Friedman(1999)].

1 choose an initial f_0, let $F_0 = f_0$

2 for $k = 1, 2, \ldots, K$

2.1 $\tilde{y}_i = -\frac{\partial L(y_i, F_{k-1}(x_i))}{\partial F_{k-1}(x_i)}$, $i = 1, 2, \ldots, N$
$\{\tilde{y}_i\}_1^N$: pseudo responses, a measurement of residuals, namely $\{y_i - F_{k-1}(x_i)\}_1^N$

2.2 $w^* = \arg\min_w \sum_{i=1}^N [\tilde{y}_i - h_k(x_i; w)]^2$
Train h_k to fit $\{(x_i, \tilde{y}_i)\}_1^N$ using square error loss

2.3 $\rho^* = \arg\min_\rho \sum_{i=1}^N L(y_i, F_{k-1}(x_i) + \rho h_k(x_i; w^*))$

2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$

3 output F_K

Gradient Boosting Machine

$\Omega(f_k) = 0$ in [Friedman(1999)].

1 choose an initial f_0, let $F_0 = f_0$

2 for $k = 1, 2, \ldots, K$

2.1 $\tilde{y}_i = -\frac{\partial L(y_i, F_{k-1}(x_i))}{\partial F_{k-1}(x_i)}$, $i = 1, 2, \ldots, N$
$\{\tilde{y}_i\}_1^N$: pseudo responses, a measurement of residuals, namely $\{y_i - F_{k-1}(x_i)\}_1^N$

2.2 $w^* = \arg\min_w \sum_{i=1}^N [\tilde{y}_i - h_k(x_i; w)]^2$
Train h_k to fit $\{(x_i, \tilde{y}_i)\}_1^N$ using square error loss

2.3 $\rho^* = \arg\min_\rho \sum_{i=1}^N L(y_i, F_{k-1}(x_i) + \rho h_k(x_i; w^*))$
Perform a line search, so that $F_{k-1} + \rho^* h_k$ reduces L most

2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$

3 output F_K

Gradient Boosting Machine

$\Omega(f_k) = 0$ in [Friedman(1999)].

1. choose an initial f_0, let $F_0 = f_0$
2. for $k = 1, 2, \ldots, K$
   - 2.1 $\tilde{y}_i = -\frac{\partial L(y_i, F_{k-1}(x_i))}{\partial F_{k-1}(x_i)}$, $i = 1, 2, \ldots, N$
     $\{\tilde{y}_i\}_1^N$: pseudo responses, a measurement of residuals, namely $\{y_i - F_{k-1}(x_i)\}_1^N$
   - 2.2 $w^* = \arg\min_w \sum_{i=1}^N [\tilde{y}_i - h_k(x_i; w)]^2$
     Train h_k to fit $\{(x_i, \tilde{y}_i)\}_1^N$ using square error loss
   - 2.3 $\rho^* = \arg\min_\rho \sum_{i=1}^N L(y_i, F_{k-1}(x_i) + \rho h_k(x_i; w^*))$
     Perform a line search, so that $F_{k-1} + \rho^* h_k$ reduces L most
   - 2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$
     Update F_k
3. output F_K

Outline

Introduction

Two aliases

- GBRT = Gradient Boosted Regression Tree
- MART = Multi Additive Regression Tree

Introduction

Two aliases

- GBRT = Gradient Boosted Regression Tree
- MART = Multi Additive Regression Tree

Usage

- Classification
- Regression
- Learning to rank

GB + DT + Square Error Loss

Let h be a DT, F be a tree ensemble.

Use square error loss

$$L(y, F) = \frac{(y - F)^2}{2} \tag{10}$$

1 choose an initial guess f_0, let $F_0 = f_0$

2 for $k = 1, 2, \ldots, K$

 2.1 $\tilde{y}_i = -\frac{\partial L(y_i, F_{k-1}(x_i))}{\partial F_{k-1}(x_i)}$, $i = 1, 2, \ldots, N$

 2.2 $w^* = \arg\min_w \sum_{i=1}^N [\tilde{y}_i - h_k(x_i; w)]^2$

 2.3 $\rho^* = \arg\min_\rho \sum_{i=1}^N L(y_i, F_{k-1}(x_i) + \rho h_k(x_i; w^*))$

 2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$

3 output F_K

GB + DT + Square Error Loss

Let h be a DT, F be a tree ensemble.

Use square error loss

$$L(y, F) = \frac{(y - F)^2}{2} \tag{10}$$

1. choose an initial guess f_0, let $F_0 = f_0$
2. for $k = 1, 2, \ldots, K$
   - 2.1 $\tilde{y}_i = y_i - F_{k-1}(x_i),\ i = 1, 2, \ldots, N$
   - 2.2 $w^* = \arg\min_w \sum_{i=1}^{N} [\tilde{y}_i - h_k(x_i; w)]^2$
   - 2.3 $\rho^* = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{k-1}(x_i) + \rho h_k(x_i; w^*))$
   - 2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$
3. output F_K

GB + DT + Square Error Loss

Let h be a DT, F be a tree ensemble.

Use square error loss

$$L(y, F) = \frac{(y - F)^2}{2} \tag{10}$$

1 choose an initial guess f_0, let $F_0 = f_0$

2 for $k = 1, 2, \ldots, K$

 2.1 $\tilde{y}_i = y_i - F_{k-1}(x_i)$, $i = 1, 2, \ldots, N$

 2.2 $\{b_j, R_j\}_1^{J*} = \arg\min_{\{b_j, R_j\}_1^J} \sum_{i=1}^N \left[\tilde{y}_i - h_k(x_i; \{b_j, R_j\}_1^J)\right]^2$

 2.3 $\rho^* = \arg\min_\rho \sum_{i=1}^N L(y_i, F_{k-1}(x_i) + \rho h_k(x_i; w^*))$

 2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$

3 output F_K

GB + DT + Square Error Loss

Let h be a DT, F be a tree ensemble.

Use square error loss

$$L(y, F) = \frac{(y - F)^2}{2} \tag{10}$$

1 choose an initial guess f_0, let $F_0 = f_0$

2 for $k = 1, 2, \ldots, K$

 2.1 $\tilde{y}_i = y_i - F_{k-1}(x_i),\ i = 1, 2, \ldots, N$

 2.2 $\{b_j, R_j\}_1^{J*} = \arg\min_{\{b_j, R_j\}_1^J} \sum_{i=1}^N \left[\tilde{y}_i - h_k(x_i; \{b_j, R_j\}_1^J)\right]^2$

 2.3 $\rho^* = \arg\min_\rho \sum_{i=1}^N \left[\tilde{y}_i - \rho h_k(x_i; \{b_j, R_j\}_1^{J*})\right]^2$

 2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$

3 output F_K

GB + DT + Square Error Loss

Let h be a DT, F be a tree ensemble.

Use square error loss

$$L(y, F) = \frac{(y - F)^2}{2} \quad (10)$$

1 choose an initial guess f_0, let $F_0 = f_0$

2 for $k = 1, 2, \ldots, K$

  2.1 $\tilde{y}_i = y_i - F_{k-1}(x_i),\ i = 1, 2, \ldots, N$

  2.2 $\{b_j, R_j\}_1^{J^*} = \arg\min_{\{b_j, R_j\}_1^J} \sum_{i=1}^N \left[\tilde{y}_i - h_k(x_i; \{b_j, R_j\}_1^J)\right]^2$

  2.3 $\rho^* = \arg\min_\rho \sum_{i=1}^N \left[\tilde{y}_i - h_k(x_i; \{\rho b_j, R_j\}_1^{J^*})\right]^2$

  2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$

3 output F_K

GB + DT + Square Error Loss

Let h be a DT, F be a tree ensemble.

Use square error loss

$$L(y, F) = \frac{(y - F)^2}{2} \tag{10}$$

1. choose an initial guess f_0, let $F_0 = f_0$
2. for $k = 1, 2, \ldots, K$
   - 2.1 $\tilde{y}_i = y_i - F_{k-1}(x_i)$, $i = 1, 2, \ldots, N$
   - 2.2 $\{b_j, R_j\}_1^{J*} = \arg\min_{\{b_j, R_j\}_1^J} \sum_{i=1}^N \left[\tilde{y}_i - h_k(x_i; \{b_j, R_j\}_1^J)\right]^2$
   - 2.3 $\rho^* = 1$
   - 2.4 let $f_k = \rho^* h_k(x; w^*)$, $F_k = F_{k-1} + f_k$
3. output F_K

GB + DT + Square Error Loss

Let h be a DT, F be a tree ensemble.

Use square error loss

$$L(y, F) = \frac{(y - F)^2}{2} \tag{10}$$

1 choose an initial guess f_0, let $F_0 = f_0$

2 for $k = 1, 2, \ldots, K$

  2.1 $\tilde{y}_i = y_i - F_{k-1}(x_i)$, $i = 1, 2, \ldots, N$

  2.2 $\{b_j, R_j\}_1^{J^*} = \arg\min_{\{b_j, R_j\}_1^J} \sum_{i=1}^N \left[\tilde{y}_i - h_k(x_i; \{b_j, R_j\}_1^J)\right]^2$

  2.3 $\rho^* = 1$

  2.4 let $f_k = h_k(x; \{b_j, R_j\}_1^{J^*})$, $F_k = F_{k-1} + f_k$

3 output F_K

Outline

Keep Trees Simple

Simpler model means less overfitting and lower variance.

Keep Trees Simple

Simpler model means less overfitting and lower variance.

Termination criteria

- # of leaves
- Tree depth
- # of instances in current node

Keep Trees Simple

Simpler model means less overfitting and lower variance.

Termination criteria

- # of leaves
- Tree depth
- # of instances in current node

Penalize splitting by $\Omega()$

$$\Omega(f) = \frac{\gamma}{2} J + \frac{\lambda}{2} \sum_{j=1}^{J} b_j^2 \tag{11}$$

γ and λ are regularization coefficient.

Subsampling/Column-Subsampling and Shrinkage

Consider GBDT + square error loss:

1 choose an initial guess f_0, let $F_0 = f_0$

2 for $k = 1, 2, \ldots, K$

  2.1 $\tilde{y}_i = y_i - F_{k-1}(x_i)$, $i = 1, 2, \ldots, N$

  2.2 $\{b_j, R_j\}_1^{J^*} = \arg\min_{\{b_j, R_j\}_1^J} \sum_{(x_i, y_i) \in \mathcal{D}} \left[\tilde{y}_i - h_k(x_i; \{b_j, R_j\}_1^J)\right]^2$

  2.3 let $f_k = h_k(x; \{b_j, R_j\}_1^{J^*})$, $F_k = F_{k-1} + f_k$

3 output F_K

Subsampling/Column-Subsampling and Shrinkage

Consider GBDT + square error loss:

1 choose an initial guess f_0, let $F_0 = f_0$

2 for $k = 1, 2, \ldots, K$

2.1 $\tilde{y}_i = y_i - F_{k-1}(x_i)$, $i = 1, 2, \ldots, N$

2.2 $\{b_j, R_j\}_1^{J^*} = \arg\min_{\{b_j, R_j\}_1^J} \sum_{(x_i', y_i) \in \mathcal{D}'} \left[\tilde{y}_i - h_k(x_i'; \{b_j, R_j\}_1^J)\right]^2$

2.3 let $f_k = h_k(x; \{b_j, R_j\}_1^{J^*})$, $F_k = F_{k-1} + f_k$

3 output F_K

Subsampling and column-subsampling

$\mathcal{D}'$ is sampled and column sampled from $\mathcal{D}$ at a sample rate.

Subsampling/Column-Subsampling and Shrinkage

Consider GBDT + square error loss:

1. choose an initial guess f_0, let $F_0 = f_0$
2. for $k = 1, 2, \dots, K$
   2.1 $\tilde{y}_i = y_i - F_{k-1}(x_i)$, $i = 1, 2, \dots, N$
   2.2 $\{b_j, R_j\}_1^{J^*} = \arg\min_{\{b_j, R_j\}_1^J} \sum_{(x_i', y_i) \in \mathcal{D}'} \left[\tilde{y}_i - h_k(x_i'; \{b_j, R_j\}_1^J)\right]^2$
   2.3 let $f_k = h_k(x; \{b_j, R_j\}_1^{J^*})$, $F_k = F_{k-1} + \eta f_k$
3. output F_K

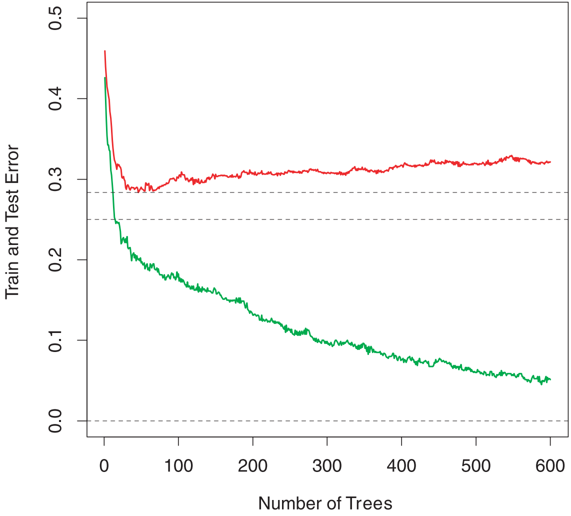
Subsampling and column-subsampling

$\mathcal{D}'$ is sampled and column sampled from $\mathcal{D}$ at a sample rate.

Shrinkage

Learning rate: η

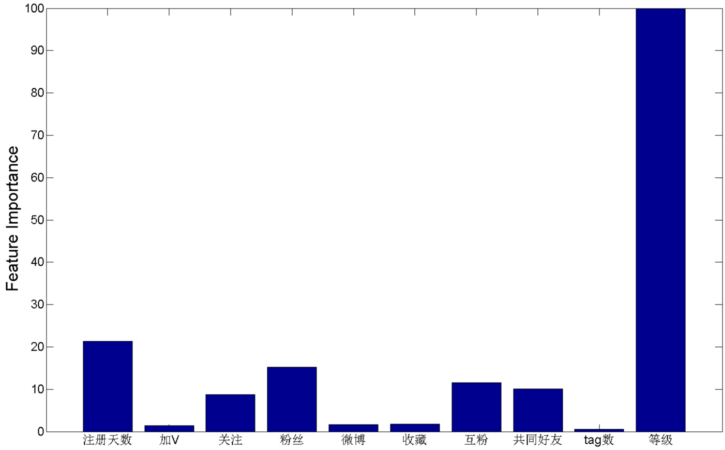
Early Stop



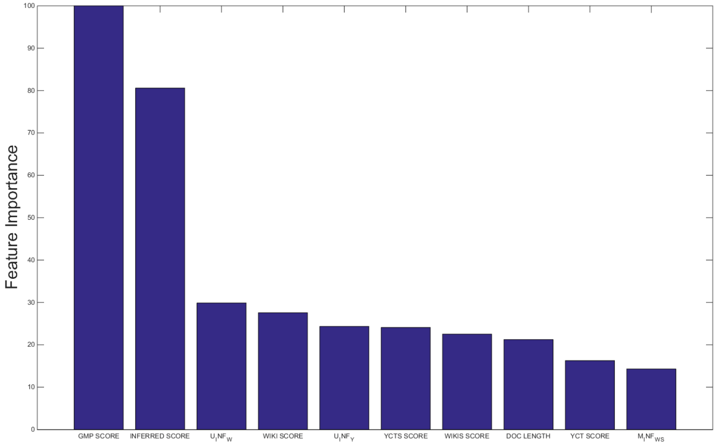
Picture from "J. Friedman, T. Hastie, R. Tibshirani(2000). Additive Logistic Regression - A Statistical View of Boosting"

Outline

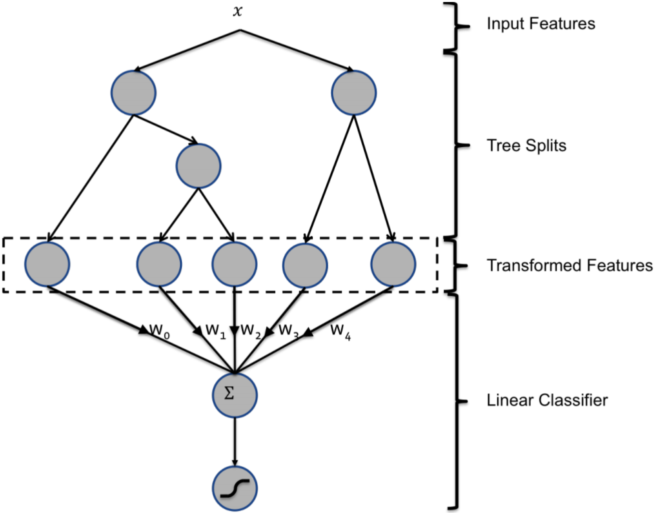
Feature Importance Demo 1



Feature Importance Demo 2

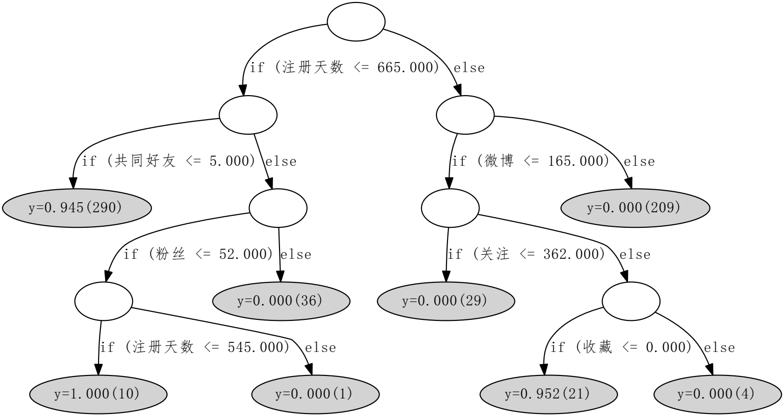


Feature Combination



Picture from "Facebook(2014). Practical Lessons from Predicting Clicks on Ads at Facebook"

A Feature Combination Demo



Outline

A Demo with xgboost

https://github.com/tqchen/xgboost

A very efficient and versatile GBDT package.

Dataset: https://archive.ics.uci.edu/ml/datasets/Mushroom

A Demo with xgboost

https://github.com/tqchen/xgboost
A very efficient and versatile GBDT package.
Dataset: https://archive.ics.uci.edu/ml/datasets/Mushroom

Dataset

A mushroom dataset from UCI.
8124 instances.
126 features.

A Demo with xgboost

https://github.com/tqchen/xgboost
A very efficient and versatile GBDT package.
Dataset: https://archive.ics.uci.edu/ml/datasets/Mushroom

Dataset

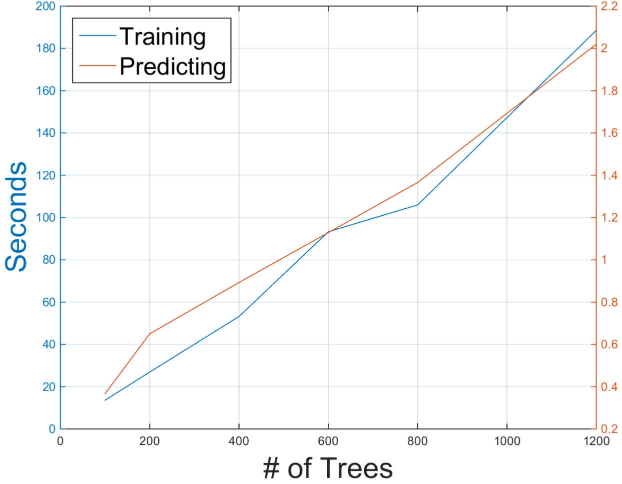
A mushroom dataset from UCI.
8124 instances.
126 features.

My default settings

$objective = binary : logistic$
$eta = 0.05$
$subsample = 0.5$
$colsample\_bytree = 1.0$
$gamma = 1.0$
$min\_child\_weight = 1$
$max\_depth = 3$
$num\_round = 100$

of Trees vs Training/Predicting Time

All tests are repeated 10 times.



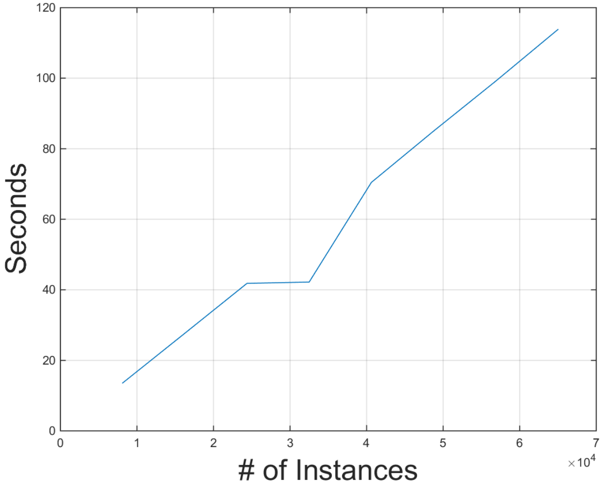
of Trees vs Training/Predicting Time Cont.

Recall that $max\_depth = 3$.

# of trees	predicting speed(instances/milli-second)
100	221
200	125
400	91
600	72
800	59

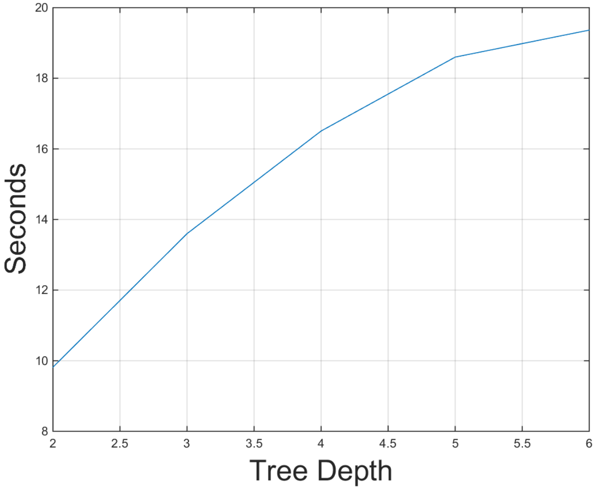
of Instances vs Training Time

All tests are repeated 10 times.



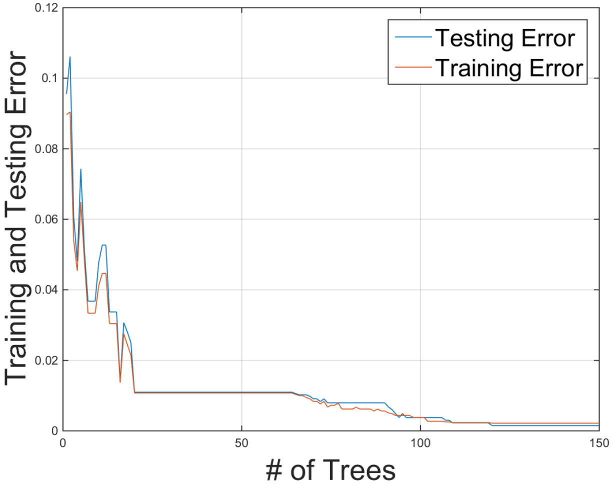
Tree Depth vs Training Time

All tests are repeated 10 times.



Training and Testing Error Curve

$subsample = 0.1$, $colsample\_bytree = 0.1$



Choose Model Parameters

Choose model parameters by validation

\# of trees: 100, 200, 400, 600, 800
tree depth: 3, 4, 5
\# of leaves: 20-50
sample rate: 0.1-0.9
learning rate: 0.01-0.1

Choose Model Parameters

Choose model parameters by validation

\# of trees: 100, 200, 400, 600, 800
tree depth: 3, 4, 5
\# of leaves: 20-50
sample rate: 0.1-0.9
learning rate: 0.01-0.1

Loss function

Square error loss and logistic loss(see Appendix) are always preferred.

Outline

GBDT vs LR

	GBDT	LR
linearity	nonlinear	linear
fitting capability	good	less good
feature selection	naturally	only l1-reg
feature combination	naturally	no
regression	yes	no
classification	yes	yes
multi-class classification	support	support
output probability	support	yes
parallelization	hard and less beneficial	easy

GBDT vs LR: Complexity

	GBDT	LR
speed	slow	fast
memory(training)	$O(N) + O(K\bar{J})$	$O(N) + O(M)$
cpu(training)	$O(K\bar{J}MN \log N)$	$O(KN)$
memory(predicting)	$O(K\bar{J})$	$O(M)$
cpu(predicting)	$O(K\bar{J})$	$O(M)$
N	large	huge
M	medium	huge

K: # of trees(GBDT) / # of iterations(LR).
J: # of leaves in one tree.
M: # of features.
N: # of instances.

Outline

Summary

- Gradient Boosting
- GBDT
- Regularization
- Feature Selection
- xgboost

Outline

Reference

J. Friedman(1999). Greedy Function Approximation: A Gradient Boosting Machine.

J. Friedman(1999). Stochastic Gradient Boosting.

J. Friedman, T. Hastie, R. Tibshirani(2000). Additive Logistic Regression - A Statistical View of Boosting.

T. Hastie, R. Tibshirani, J. Friedman(2008). Chapter 10 of The Elements of Statistical Learning(2e).

Outline

Square and Absolute Error

Square error

$$L(y, F) = \frac{(y - F)^2}{2} \tag{12}$$

$$\tilde{y} = y - F \tag{13}$$

Absolute error

$$L(y, F) = |y - F| \tag{14}$$

$$\tilde{y} = sign(y - F) \tag{15}$$

Logistic Loss and LogitBoost

For binary classification, $y \in \{-1, 1\}$.

Logistic loss

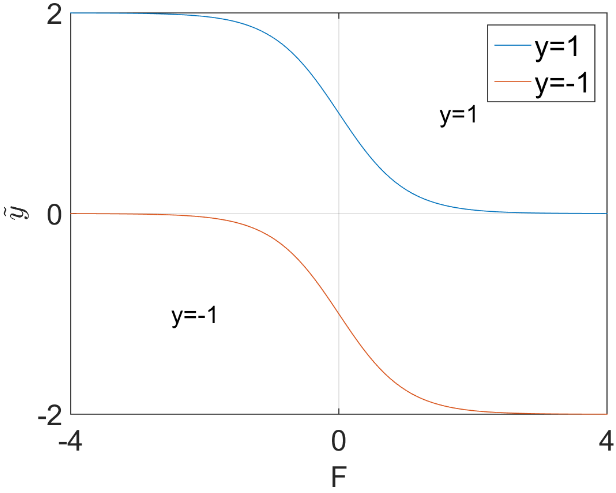
$$L(y, F) = \log(1 + \exp(-2yF)) \tag{16}$$

$$\tilde{y} = \frac{2y}{1 + \exp(2yF)} \tag{17}$$

This will result in the LogitBoost algorithm.

Logistic Loss and LogitBoost Cont.

For logistic loss:



Exponential Loss and Adaboost

For binary classification, $y \in \{-1, 1\}$.

Exponential loss

$$L(y, F) = \exp(-yF) \tag{18}$$

$$\tilde{y} = y \exp(-yF) \tag{19}$$

This will result in the Adaboost algorithm.

Outline

Deduction: Taylor Series of L

Continued from page (19).

$$
\begin{aligned}
L(y_i, F_k(x_i; w)) &= L(y_i, F_{k-1}(x_i; w) + f_k(x_i)) && (20)\\
\approx L(y_i, F_{k-1}(x_i; w)) &+ \underbrace{\frac{\partial L(y_i, F_{k-1}(x_i; w))}{\partial F_{k-1}}}_{:=g_i} f_k(x_i) \\
&+ \frac{1}{2} \underbrace{\frac{\partial^2 L(y_i, F_{k-1}(x_i; w))}{\partial F_{k-1}^2}}_{:=h_i} f_k^2(x_i) && (21)\\
= L(y_i, F_{k-1}(x_i; w)) &+ g_i f_k(x_i) + \frac{1}{2} h_i f_k^2(x_i) && (22)
\end{aligned}
$$

Deduction: $\mathcal{L}$ and Ω

Loss function with respect to f_k

$$\mathcal{L}(f_k) = \sum_{i=1}^{N} \left[g_i f_k(x_i) + \frac{1}{2} h_i f_k^2(x_i) \right] + \Omega(f_k) \tag{23}$$

From now on, we assume f is a decision tree.

A common $\Omega()$ for decision tree f

$$\Omega(f) = \frac{\gamma}{2} J + \frac{\lambda}{2} \sum_{j=1}^{J} b_j^2 \tag{24}$$

γ and λ are regularization coefficient.

Deduction: $\{b_j\}_1^J$

With $\{R_j\}_1^J$ known, we are to optimize leaf values $\{b_j\}_1^J$.
How to get $\{R_j\}_1^J$ will be described later.

The optimal leaf values are given by minimizing $\mathcal{L}$

$$\mathcal{L}(f_k) = \mathcal{L}(\{b_j\}_1^J, \{R_j\}_1^J) = \sum_{j=1}^{J} \left[\underbrace{\sum_{x_i \in R_j} g_i}_{:=G_j} b_j + \frac{1}{2} \left(\underbrace{\sum_{x_i \in R_j} h_i}_{:=H_j} + \lambda \right) b_j^2 \right] + \frac{\gamma}{2} J \tag{25}$$

To keep symbols uncluttered, J, b_j and R_j are parameters of f_k.

Deduction: $\{b_j\}_1^J$ Cont.

The optimal leaf value of R_j

$$b_j^* = \arg\min_{b_j} \mathcal{L}(\{b_j\}_1^J, \{R_j\}_1^J) = -\frac{G_j}{H_j + \lambda} \tag{26}$$

The minimal loss function with $\{R_j\}_1^J$ fixed

$$\mathcal{L}(\{b_j^*\}_1^J, \{R_j\}_1^J) = -\frac{1}{2}\sum_{j=1}^{J} \frac{G_j^2}{H_j + \lambda} + \frac{\gamma}{2} J \tag{27}$$

Deduction: $\{R_j\}_1^J$

With $\{R_j\}_1^J$ and $\{b_j^*\}_1^J$, now we consider splitting R_i to R_L and R_R.

Define $\mathbf{R}$ and $\mathbf{R}'$: old and new splitting scheme

$$\mathbf{R} : \{R_j\}_1^J \tag{28}$$

$$\mathbf{R}' : \{R_j\}_{1, j \neq i}^J, R_L, R_R \tag{29}$$

Define $\mathbf{b}^*$ and $\mathbf{b}^{*\prime}$

$$\mathbf{b}^* := \{b_j^*\}_1^J \tag{30}$$

$$\mathbf{b}^{*\prime} := \{b_j^*\}_{1, j \neq i}^J + \{b_L^*, b_R^*\} \tag{31}$$

Deduction: $\{R_j\}_1^J$ Cont.

Gain is defined to be the decrease in $\mathcal{L}$ after splitting R_i, and use (27).

Gain

$$Gain = 2\mathcal{L}(\mathbf{b}^*, \mathbf{R}) - 2\mathcal{L}(\mathbf{b}^{*'}, \mathbf{R'}) \tag{32}$$

$$= \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma \tag{33}$$

Terms of 3 colors:

- Gain from splitting
- Gain from not splitting
- The complexity introduced by splitting

The Tree Building Algorithm

(33) directly conducts us to split

- Calculate g_i and h_i for all instances
- Iterate among all features
  - Iterate among all sorted values of current feature
    - Calculate gain after splitting at current value
- Select the "best" splitting scheme, which results in the maximal gain
- Stop splitting if the maximal gain is negative

Outline

GBDT + Square Error Loss

$$L(y, F) = \frac{(y - F)^2}{2} \tag{34}$$

$$g_i = F_{k-1}(x_i) - y_i = -\tilde{y}_i \tag{35}$$

$$h_i = 1 \tag{36}$$

And set λ and γ zero, we have

$$b_j^* = -\frac{\sum_{x_i \in R_j} g_i}{\sum_{x_i \in R_j} h_i} = \frac{\sum_{x_i \in R_j} \tilde{y}_i}{\sum_{x_i \in R_j} \mathbf{1}} \tag{37}$$

This is completely equivalent to LSBoost(23)[Friedman(1999)].

Outline

GBDT + Logistic Loss

$$L(y, F) = \log(1 + \exp(-2yF)) \tag{38}$$

$$g_i = -\frac{2y_i}{1 + \exp(2y_i F_{k-1}(x_i))} \tag{39}$$

$$h_i = \frac{4\exp(2y_i F_{k-1}(x_i))}{[1 + \exp(2y_i F_{k-1}(x_i))]^2} = |g_i|\,(2 - |g_i|) \tag{40}$$

And set λ and γ zero, we have

$$b_j^* = -\frac{\sum_{x_i \in R_j} g_i}{\sum_{x_i \in R_j} |g_i|\,(2 - |g_i|)} \tag{41}$$

$$p(y = 1|x) = \frac{1}{1 + \exp^{-2F(x)}} \tag{42}$$

This is completely equivalent to LogitBoost[Friedman(1999)].