电子科技大学
University of Electronic Science and Technology of China

XGBoost: A Version of Gradient Boosting Tree

Heng Zhang

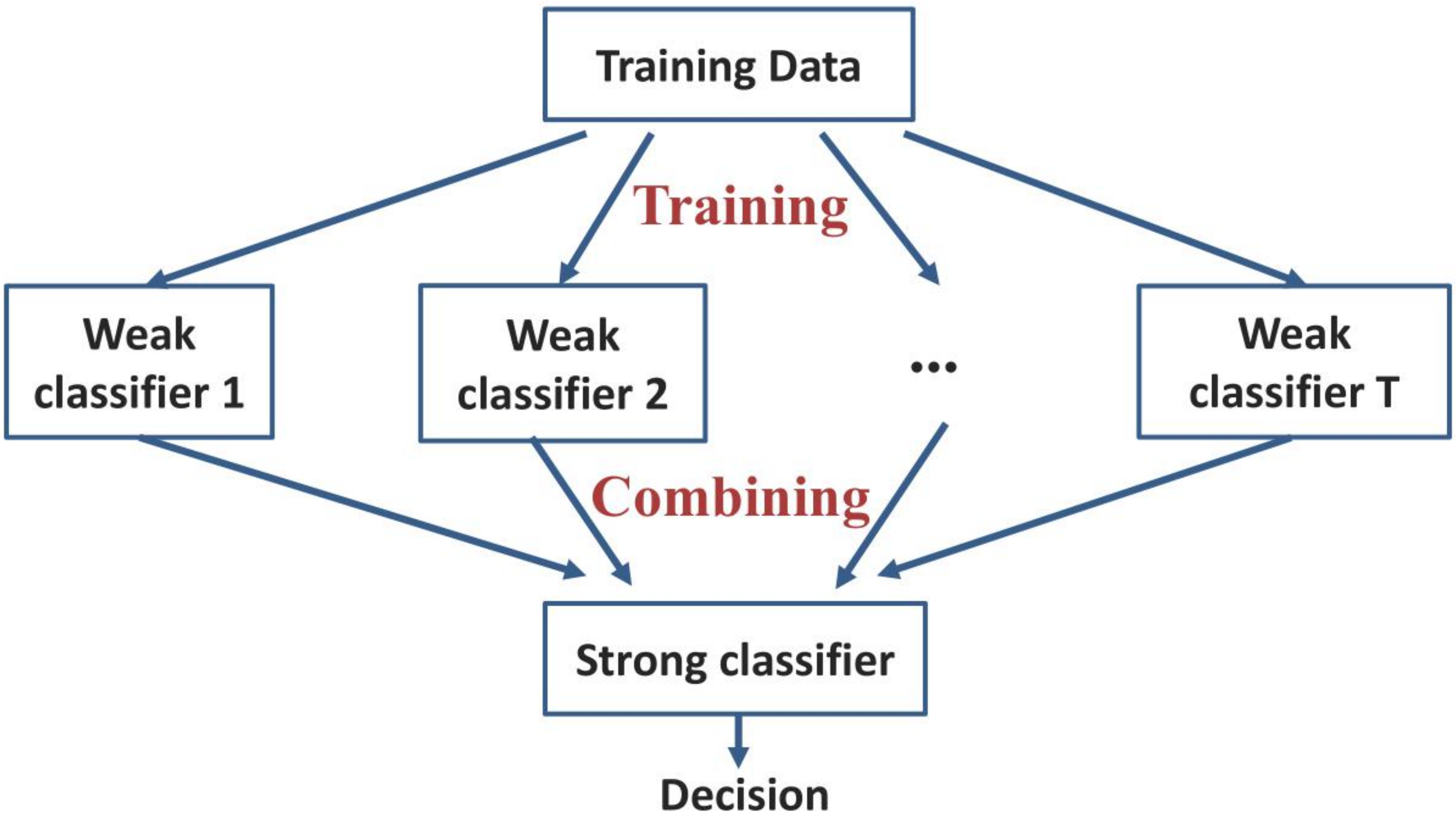
Data Mining Lab,
Big Data Research Center, UESTC
Email： hengzhang64@gmail.com

Outline

- Review of Ensemble Learning
- Regression Tree and Ensemble
- Gradient Boosting
- Summary

- Ensemble learning

Multiple classifiers are trained and combined to solve a same problem.



Why Ensemble Learning

数据挖掘实验室
Data Mining Lab

- Suppose we have 5 completely independent classifiers for majority voting

- If accuracy is 70% for each

  10 (.7^3)(.3^2)+5(.7^4)(.3)+(.7^5)

  83.7% majority vote accuracy

- 101 such classifiers

  99.9% majority vote accuracy

Bagging

Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

The number of base classifiers

Process:

for $t = 1, \cdots, T$:

Complete name of bagging is Bootstrap aggregating

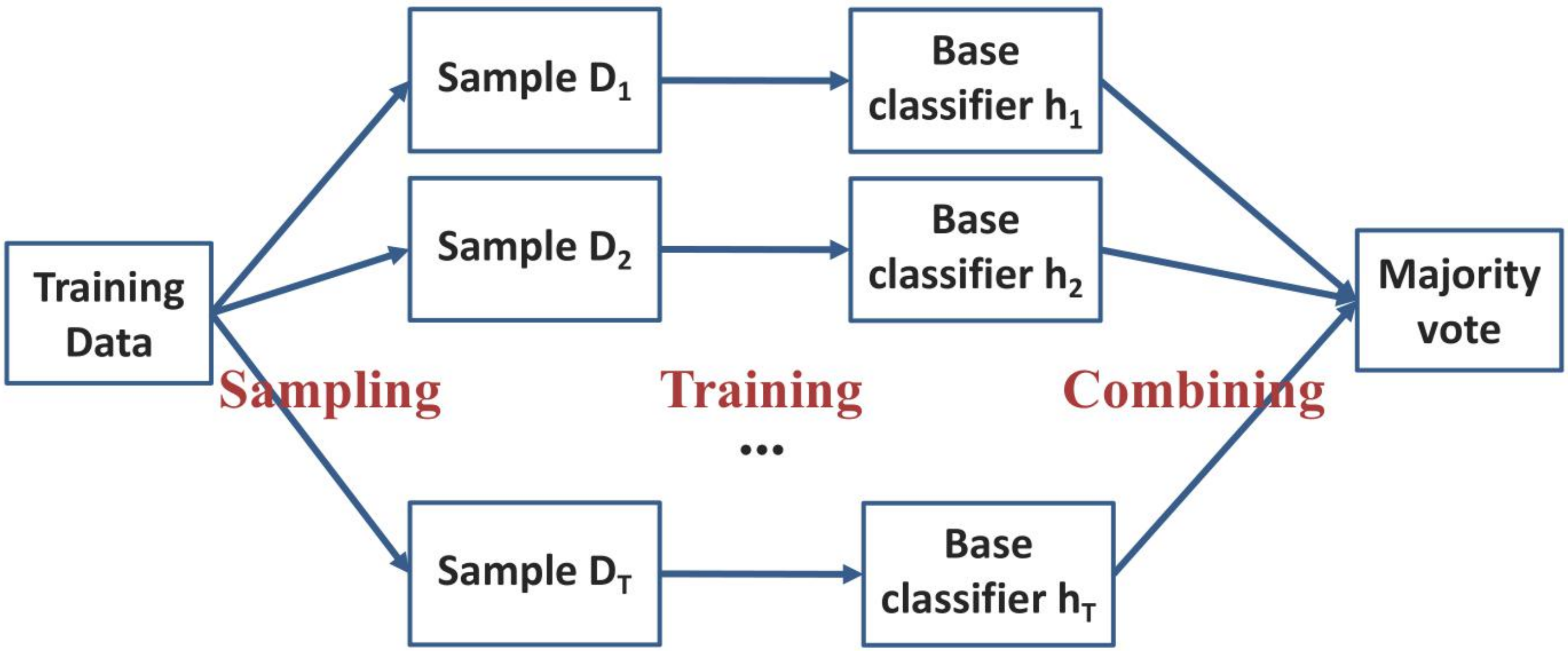
$\mathcal{D}_t = Bootstrap(\mathcal{D})$; % Generate a bootstrap sample from $\mathcal{D}$
$h_t = \mathfrak{L}(\mathcal{D}_t)$ % Train a base learner h_t from the bootstrap sample

end.

Output: $H(\boldsymbol{x}) = \text{argmax}_{y \in \mathcal{Y}} \sum_{t=1}^{T} 1(y = h_t(\boldsymbol{x}))$ % the value of $1(a)$ is 1 if a is $true$ and 0 otherwise

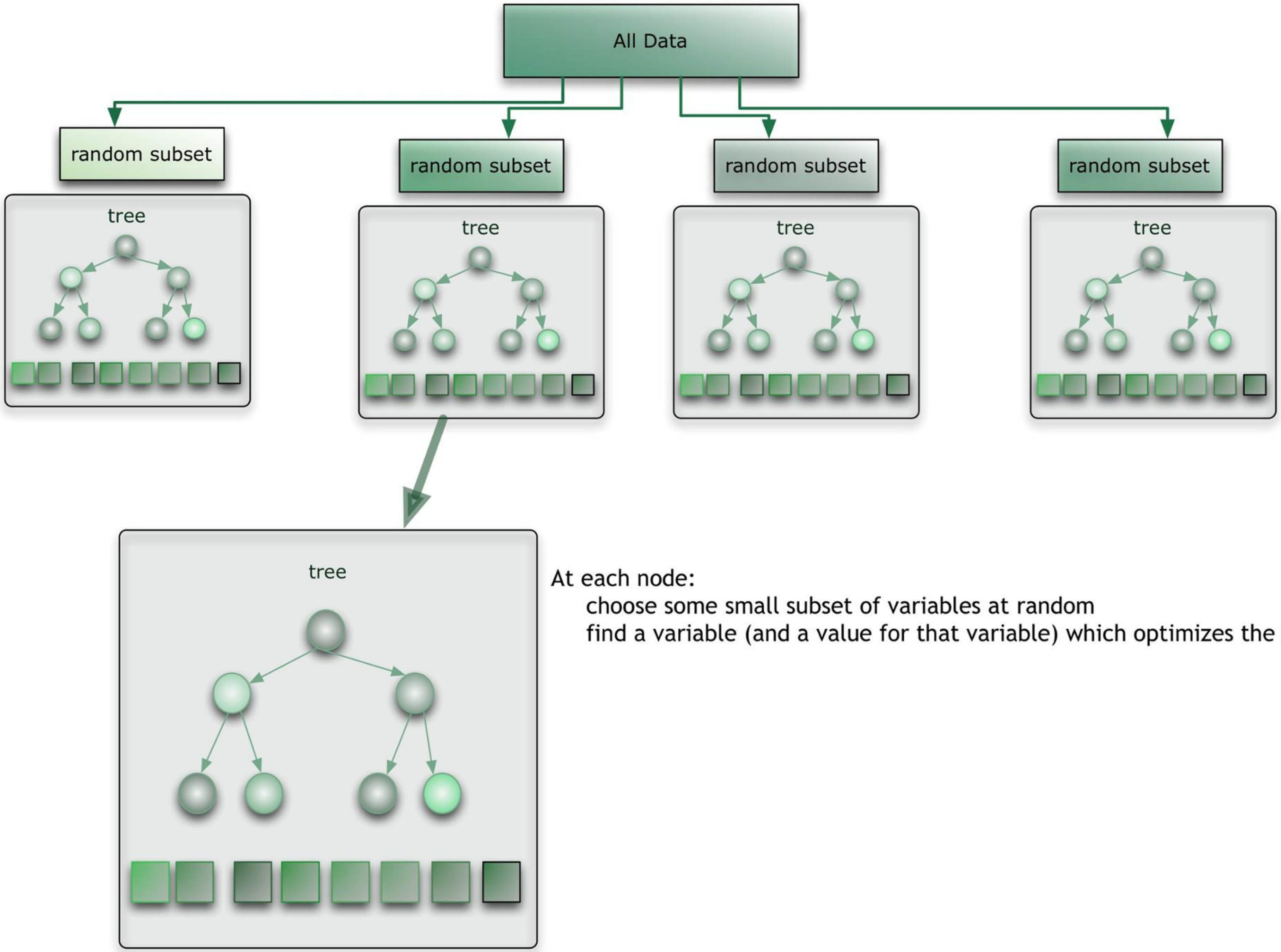
Find the most-voted class

Bagging



- Samples are independent and replacement (有放回)
- Base classifiers can be generated in a parallel style
  — Save time

Random Froest



At each node:

choose some small subset of variables at random

find a variable (and a value for that variable) which optimizes the split

Boosting

Main idea:

Learn the examples with high error rate intensively

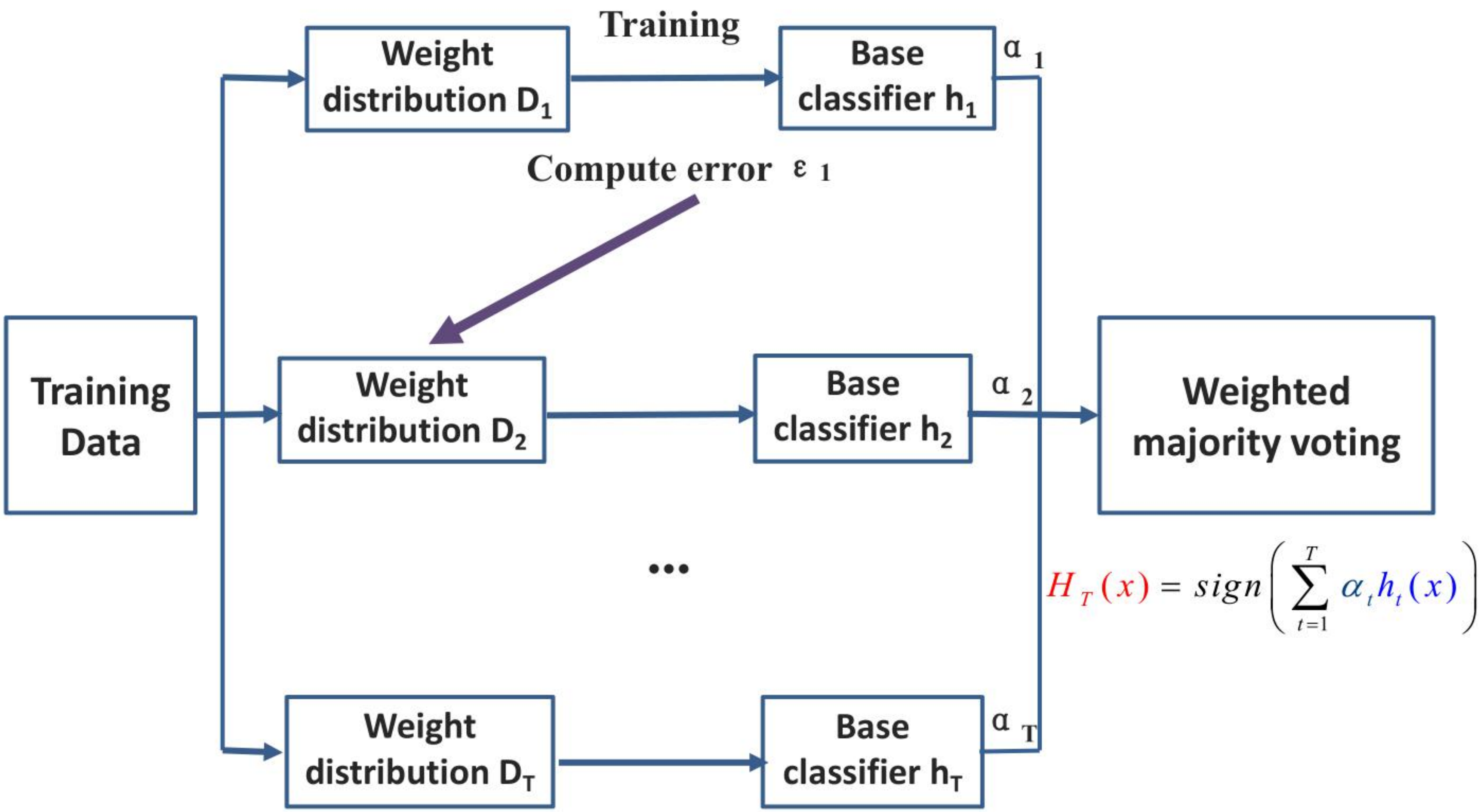
Training:

- Assign equal weights (probabilities) to all the training examples
- Train a base learner from the training data set
- Test it , and update the weights (Increasing the weights of incorrectly classified examples)
- Train next classifier from updated weight distribution(consider more about incorrect examples), repeat for T times

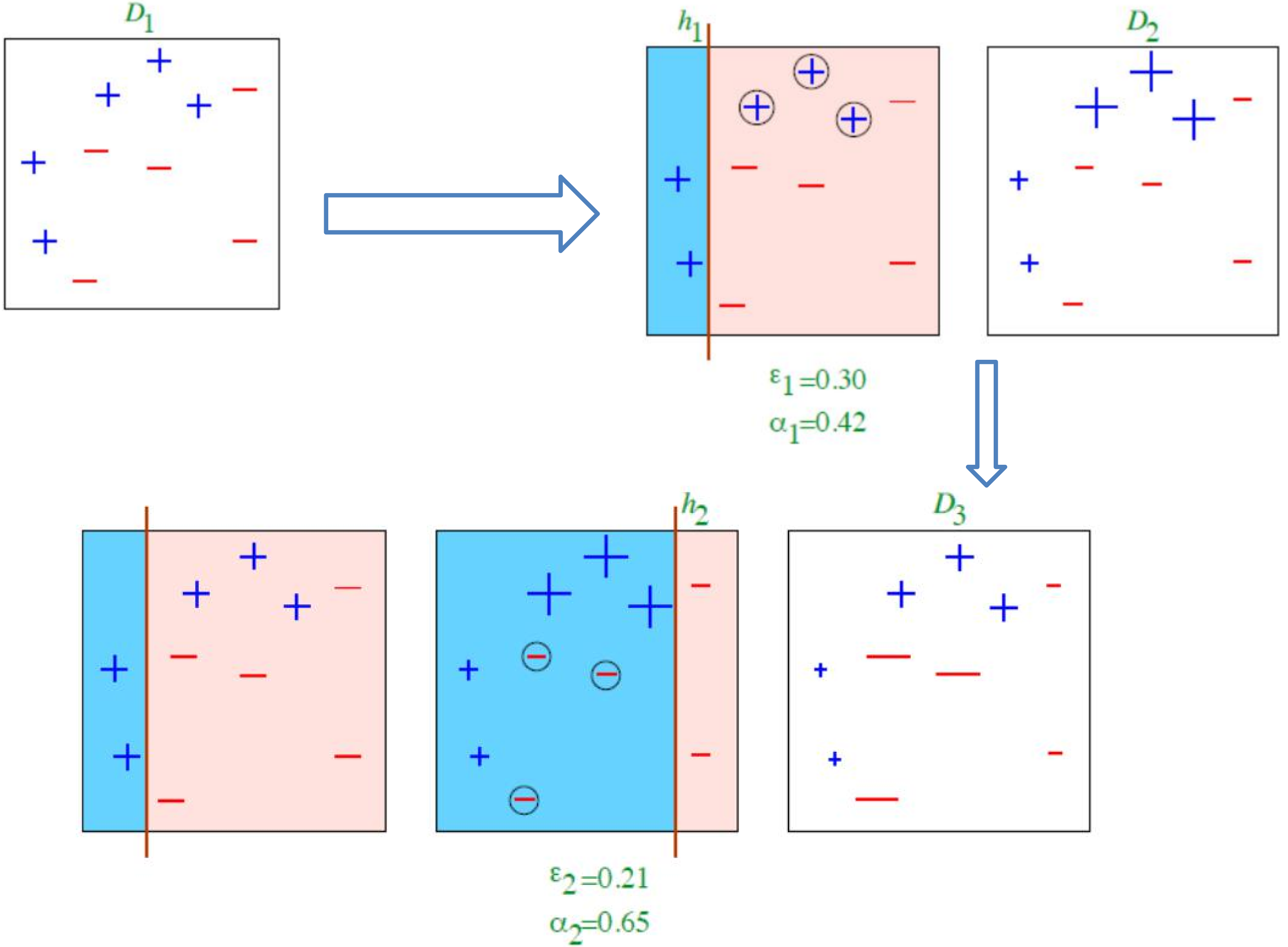
Combining:

— Weighted majority voting (linear combination)

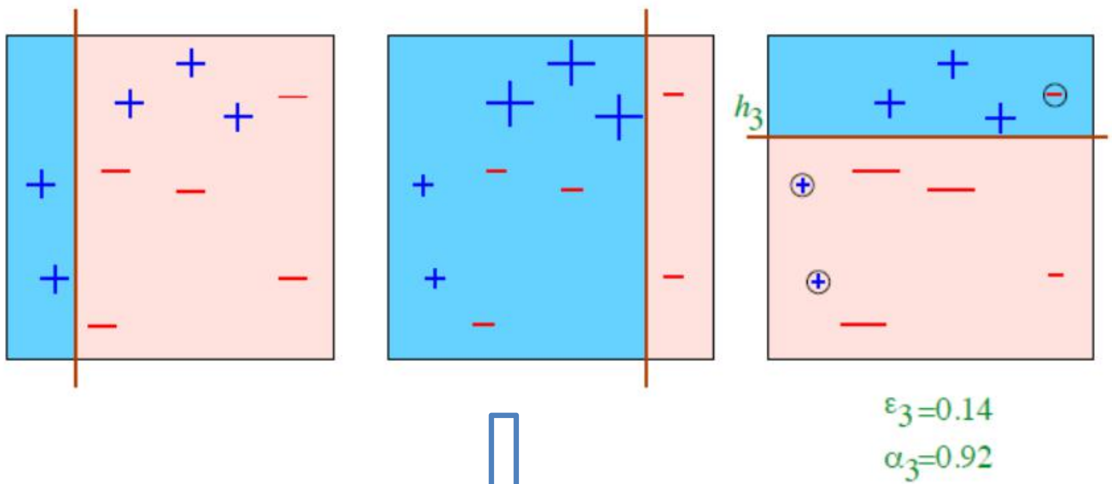
Boosting


Training
Weight distribution D_1
Base classifier h_1
α_1
Compute error ε_1
Training Data
Weight distribution D_2
Base classifier h_2
α_2
Weighted majority voting
…
Weight distribution D_T
Base classifier h_T
α_T
$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

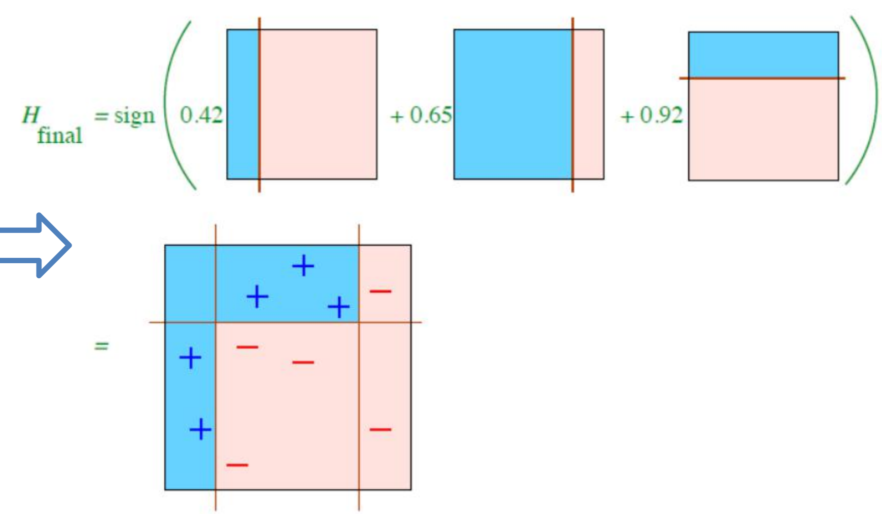
Adaboost:


D_1
h_1
D_2
ε_1=0.30
α_1=0.42
h_2
D_3
ε_2=0.21
α_2=0.65

Adaboost:



$$H(x) = \sum_t \rho_t h_t(x)$$



- Notations: $x_i \in R^d$ i-th training example
- **Model**: how to make prediction $\hat{y}_i$ given x_i

  e.g Linear model: $\hat{y}_i = \sum_j w_j x_{ij}$

  The prediction score $\hat{y}_i$ can have different interpretations depending on the task

  Regression: $\hat{y}_i$ is the predicted score

  Classification: $\hat{y}_i$ is the probability of the instance being on class

  Others... for example in ranking $\hat{y}_i$ can be the rank score
- **Parameters** : the things we need to learn from data

  Linear model: $\Theta = \{\mathrm{w}_j \mid j = 1, \ldots, d\}$

Objective Function

- Objective function that is everywhere

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

Training Loss measures how well model fit on training data

Regularization measures complexity of model

- Loss on training data: $L = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$

Square loss: $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$

Logistic loss: $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i)\ln(1 + e^{\hat{y}_i})$

- Regularization: how complicated the model is?

L2 norm: $\Omega(w) = \lambda \|w\|^2$

L1 norm(lasso): $\Omega(w) = \lambda \|w\|_1$

Putting them in to context

- Ridge regression: $\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda\|w\|^2$

  Linear model, square loss, L2 regularization
- Lasso: $\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda\|w\|_1$

  Linear model, square loss, L1 regularization
- Logistic regression:

$$\sum_{i=1}^{n}[y_i \ln(1+e^{-w^T x_i}) + (1-y_i)\ln(1+e^{w^T x_i})] + \lambda\|w\|^2$$

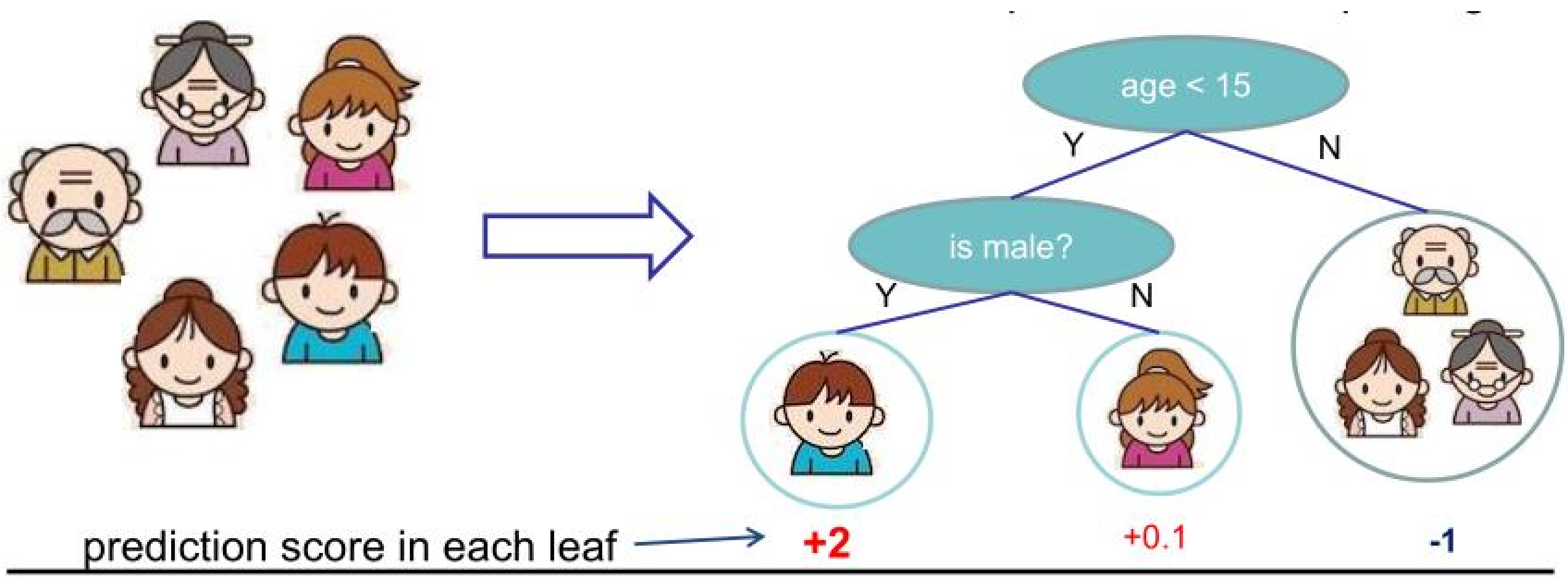
  Linear model, logistic loss, L2 regularization

- ....

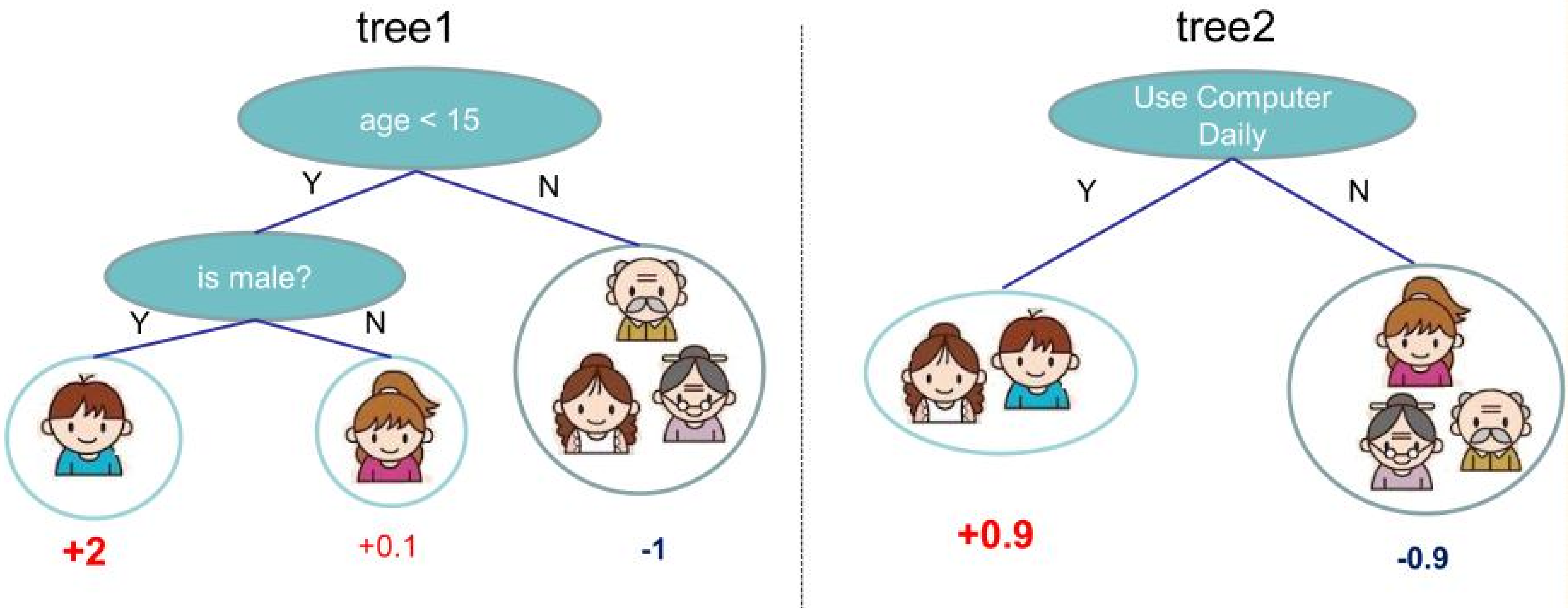
Regression Tree

- Regression tree
  - Decision rules same as decision tree
  - Contains one score in each leaf value

Input： age, gender, occupation,… Does the person like computer games?



Ensemble Tree



f() = 2 + 0.9= 2.9 f()= -1 - 0.9= -1.9

Prediction of is sum of scores predicted by each of the tree

Put into context:Model and Parameters

Model: assuming we have K trees

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

Space of functions containing all Regression trees

Objective:

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training Loss

Complexity of the Trees

Additive Training

- We can not use methods such as SGD, to find f (since they are trees, instead of just numerical vectors)
- Start from constant prediction. add a new function each time

$$\hat{y}_i^{(0)} = 0$$

$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$

$$y_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$

$$\cdots$$

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

Model at training round *t*

Keep functions added in previous round

New function

Gradient for what?

$$f_t(x_1) = y_1 - F_{t-1}(x_1)$$

$$f_t(x_2) = y_2 - F_{t-1}(x_2)$$

$$\cdots$$

$$f_n(x_n) = y_n - F_{t-1}(x_n)$$

- Just fit a regression tree f_t to data:

$$\{(x_1, y_1 - F_{t-1}(x_1)), (x_2, y_2 - F_{t-1}(x_2)), \ldots, (x_n, y_n - F_{t-1}(x_n))\}$$

- $y_i - F_{t-1}(x_i)$ are called **residuals**
- is **the direction of global optimal i.e. Gradient**

Additive Training

数据挖掘实验室

Data Mining Lab

So， how do we decide which f to add?

The prediction at round t is $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$

This is what we need to decide in round t

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i)$$

$$= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

Goal: find f_t to minimize this

Take Taylor expansion of the objective **Recall:**

$$f(x+\Delta x) = f(x) + f^{'}(x)\Delta x + \frac{1}{2} f^{''}(x)\Delta x^2 + \cdots + R_n(x)$$

Additive Training

Define:

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

If you are not comfortale with this, take the square loss as an example:

$$g_i = \partial_{\hat{y}^{(t-1)}} (\hat{y}^{(t-1)} - y_i)^2 = 2(\hat{y}^{(t-1)} - y_i)$$

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

Define Complexity of a Tree

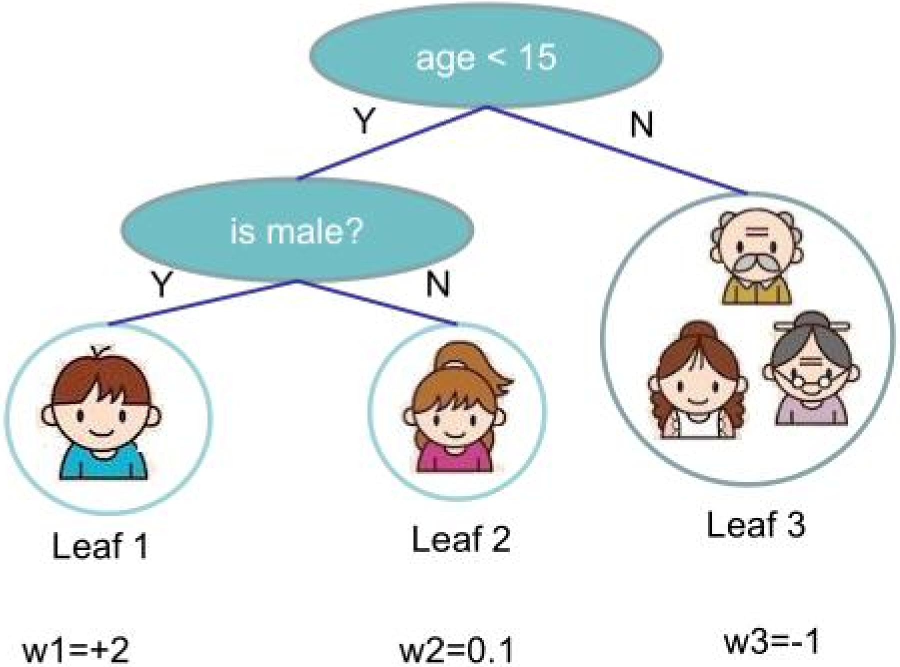
数据挖掘实验室
Data Mining Lab

Define complexity as (this is not the only possible definition)

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

Number of leaves **L2 norm of leaf scores**

For Example:



$$\Omega = \gamma \cdot 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

Revisit Objectives

数据挖掘实验室
Data Mining Lab

Define the instance set in leaf j as $I_j = \{i \mid q(x_i) = j\}$

Regroup the objective by each leaf:

$$
\begin{aligned}
Obj^{(t)} &\approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \tfrac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \\
&= \sum_{i=1}^{n} \left[g_i w_{q(x_i)} + \tfrac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \lambda \tfrac{1}{2} \sum_{j=1}^{T} w_j^2 \\
&= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i\right) w_j + \tfrac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda\right) w_j^2 \right] + \gamma T
\end{aligned}
$$

This is sum of T independent quadratic functions

The Structure Score

Let us define $G_j = \sum_{i \in I_j} g_i \quad H_j = \sum_{i \in I_j} h_i$

Then

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

$$= \sum_{j=1}^{T} \left[G_j w_j + \boxed{\tfrac{1}{2}(H_j + \lambda)} w_j^2 \right] + \gamma T$$

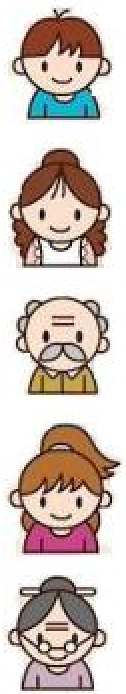
大于0

Assume the structure of tree (q(x)) is fixed, the optimal weight in each leaf, and the resulting objective value are：

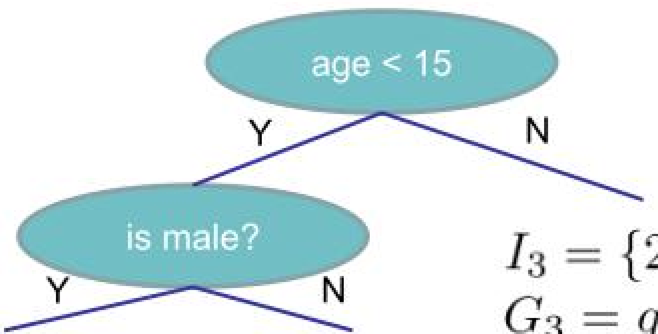
$$w_j^* = -\tfrac{1}{2} \frac{G_j^{\ 2}}{H_j + \lambda} \qquad Obj = -\tfrac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

The Structure Score Calculation

The Demo



Instance index	gradient statistics
1	g1, h1
2	g2, h2
3	g3, h3
4	g4, h4
5	g5, h5



$I_1 = \{1\}$
$G_1 = g_1$
$H_1 = h_1$

$I_2 = \{4\}$
$G_2 = g_4$
$H_4 = h_4$

$I_3 = \{2, 3, 5\}$
$G_3 = g_2 + g_3 + g_5$
$H_3 = h_2 + h_3 + h_5$

$$Obj = -\sum_j \frac{G_j^2}{H_j+\lambda} + 3\gamma$$

The smaller the score is, the better the structure is

Greedy Learning of the Tree

数据挖掘实验室
Data Mining Lab
LESS IS MORE

Until now, there find the best tree structure, and get the optimal leaf score

However ... there can be infinite possible tree structures

In practice, greedy strategy is the best to find tree structure

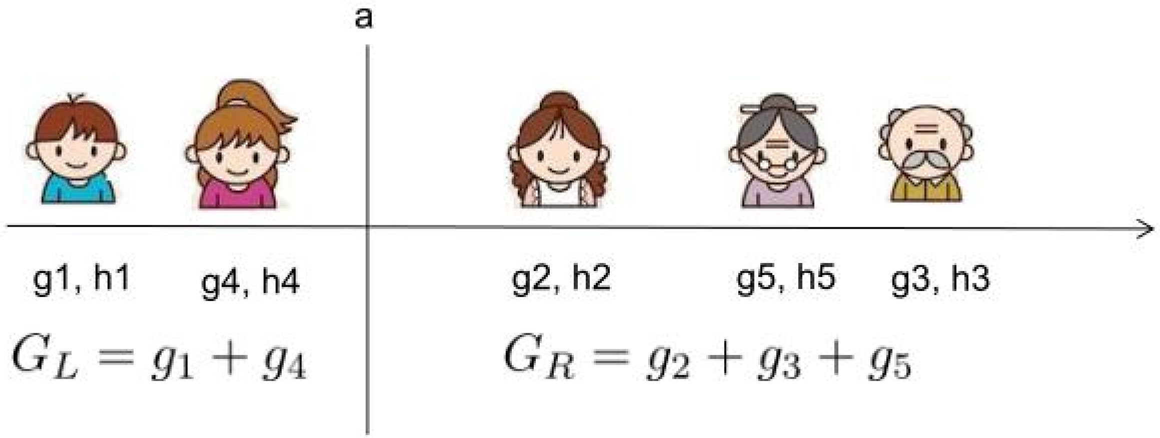
Start from tree with depth 0
For each leaf node of the tree, try to add a split. The change of objective after adding the split is

$$Gain = \frac{1}{2}\left[\underbrace{\frac{G_L^2}{H_L+\lambda}}_{\text{The score of left child}} + \underbrace{\frac{G_R^2}{H_R+\lambda}}_{\text{The score of right child}} - \underbrace{\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}}_{\text{The score of if we do not split}}\right] - \underbrace{\gamma}_{\text{The complexity cost by introducing additional leaf}}$$

Efficient Finding of the Best Split

Remaining question: how do we find the best split?

What is the gain of a split rule $x_j < a$? Say x_j is age



All we need is sum of g and h in each side, and calculate

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

Left to right linear scan over sorted instance is enough to decide the best split along the feature

Pruning and Stopping

Recall the gain of split, it can be negative!

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

- When the **training loss reduction** is smaller than **regularization**
- Trade-off between simplicity and predictivness

Stepwise and Summary

数据挖掘实验室
Data Mining Lab

- Add a new tree in each iteration
- Begining of each iteration, calaulate

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

- Use the statistics to greedily grow a tree $f_t(x)$

$$Obj = -\frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

- Add $f_t(x)$ to the model $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$
  - Usually, instead we do $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \varepsilon \cdot f_t(x_i)$
  - ε is called step-wise or shrinkage, set (0,1]
  - This means do not do full optimization in each step and reserve chance for future rounds, it helps prevent overfitting

Thanks