Gradient Boosting

Jian Li
李建

Xgboost

- Scalable, Portable and Distributed Gradient Boosting (GBDT, GBRT or GBM) Library, for Python, R, Java, Scala, C++ and more.

- https://github.com/dmlc/xgboost

- 基于论文

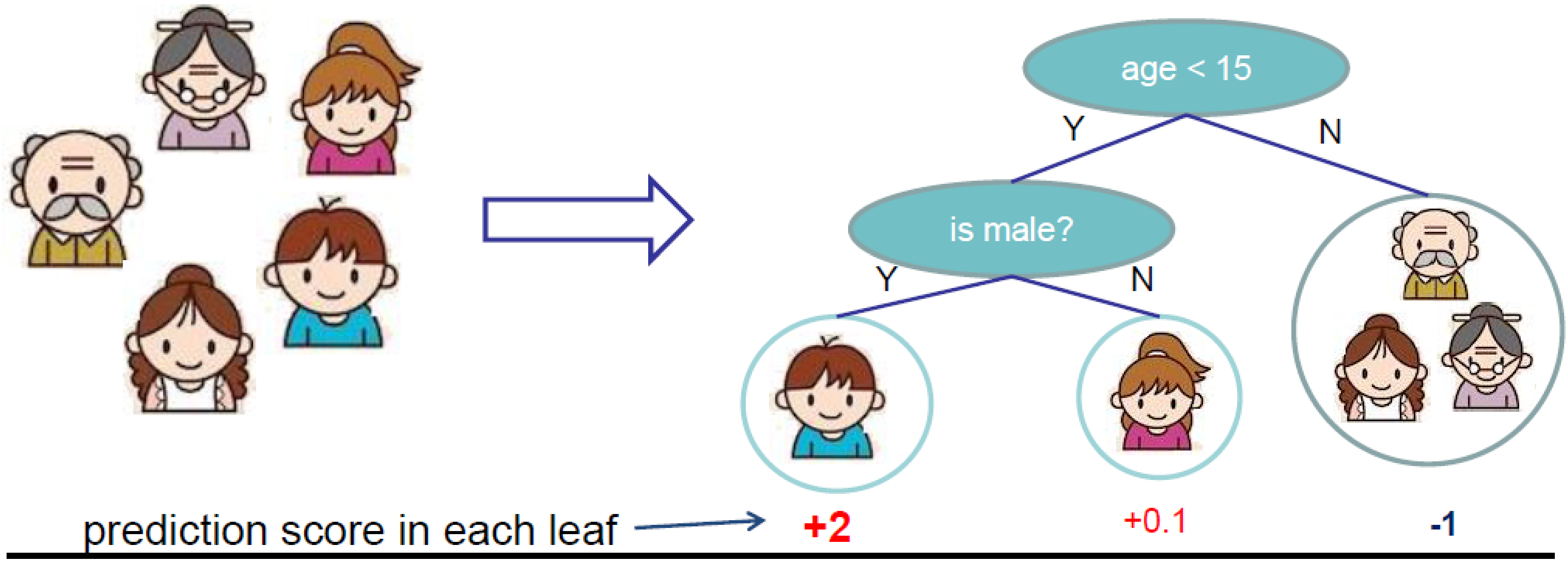
XGBoost: A Scalable Tree Boosting System

regression tree

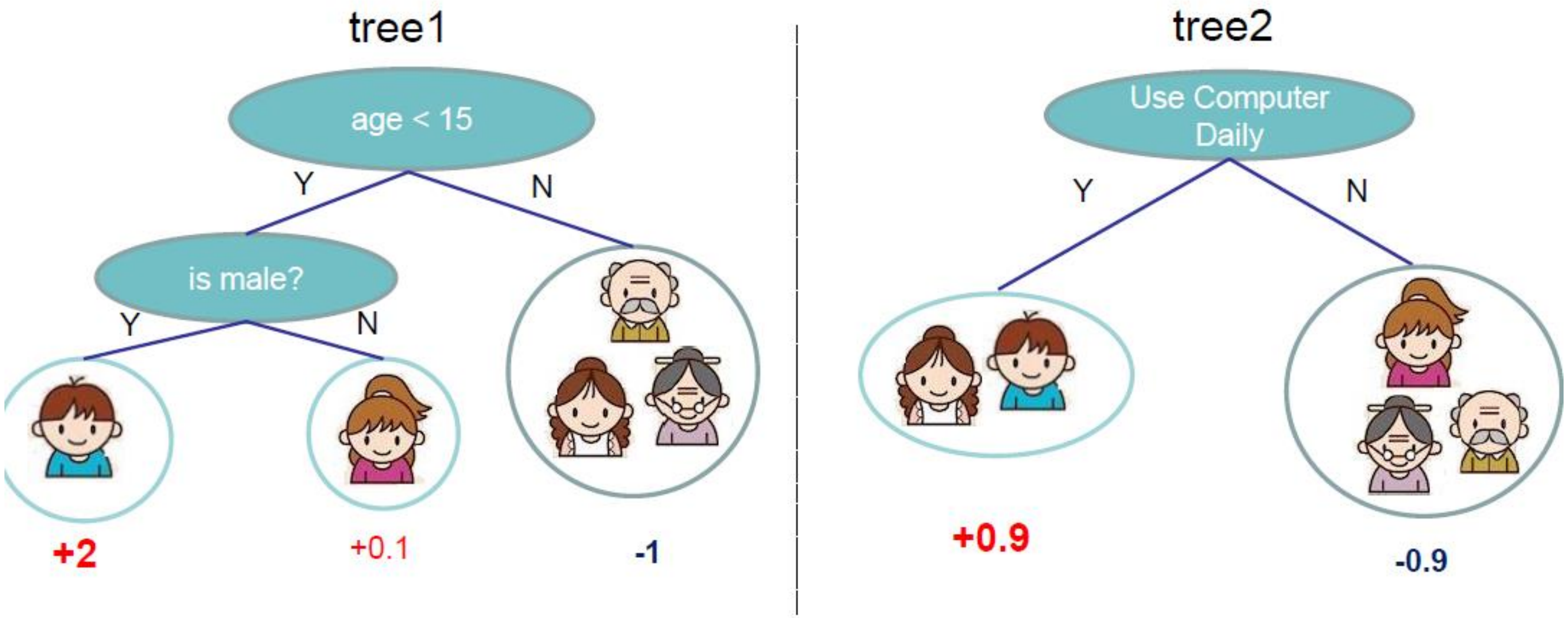
- CART (Classification And **Regression Trees**)

Input: age, gender, occupation, …

Does the person like computer games



Ensemble of regression 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
预测的分数是每个树的预测分数的和

主要用于Gradient Boosting和random forest

Ensemble of regression tree

- Suppose there are k trees 假设有K个树，f_k是第k个树定义的函数

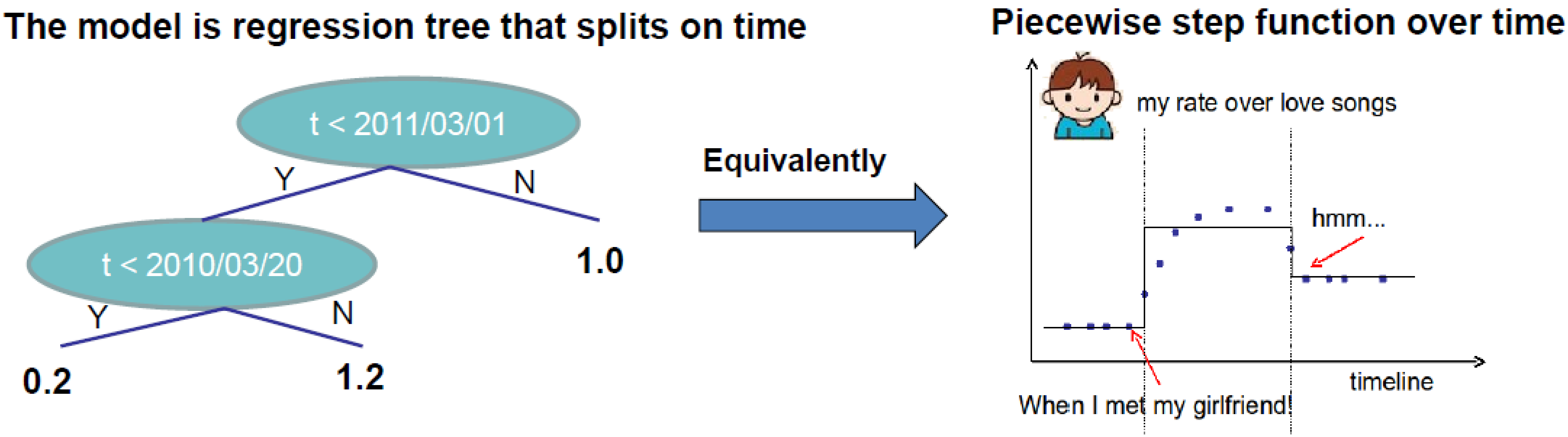
$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

$\boxed{\text{Space of functions containing all Regression trees}}$

所有regression tree定义的函数的集合

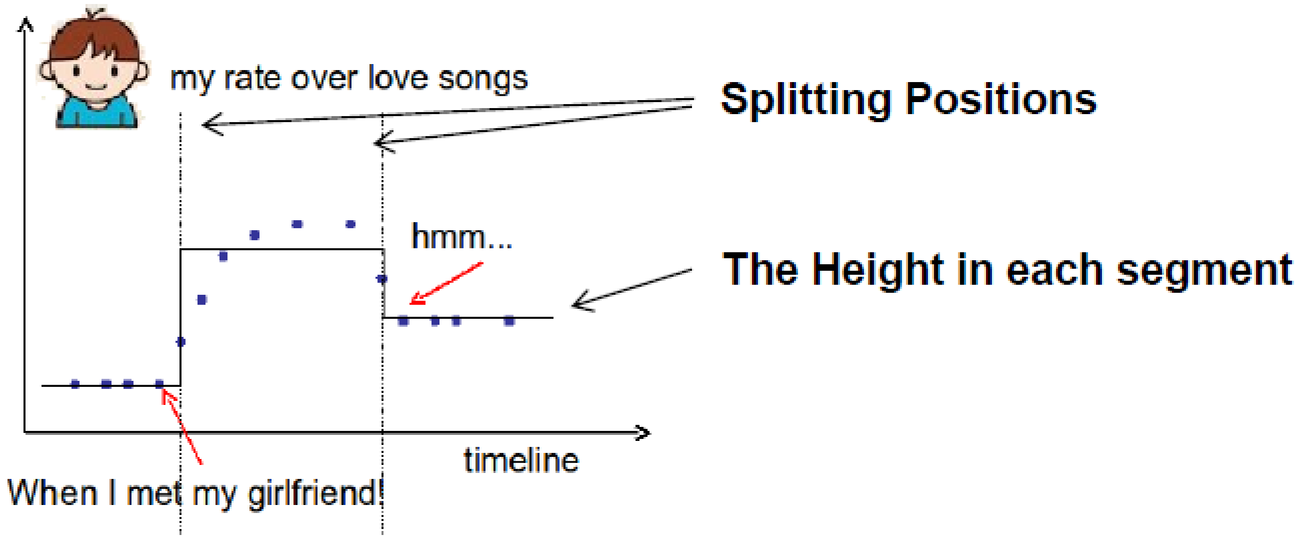
如何学习一棵树

- Example:
  - Consider regression tree on single input t (time)
  - I want to predict whether I like romantic music at time t



如何学习一棵树

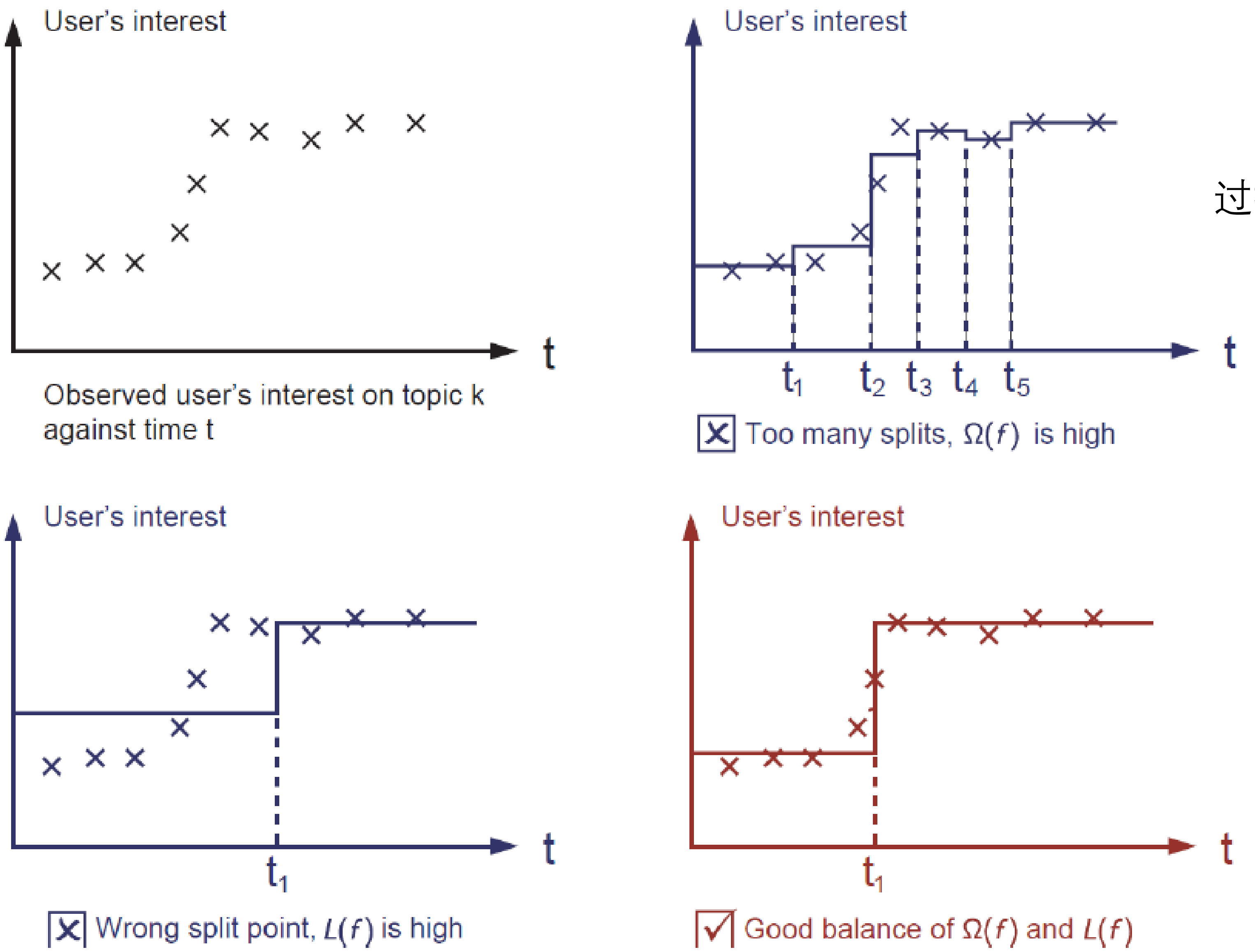
- 用分段函数近似原函数



Objective for single variable regression tree(step functions)　目标函数

- Training Loss: How will the function fit on the points?　训练的损失函数
- Regularization: How do we define complexity of the function?　正则项
    - Number of splitting points, l2 norm of the height in each segment?

过拟合


User's interest
t
Observed user's interest on topic k against time t
User's interest
t
t_1 t_2 t_3 t_4 t_5
Too many splits, $\Omega(f)$ is high
User's interest
t
t_1
Wrong split point, $L(f)$ is high
User's interest
t
t_1
Good balance of $\Omega(f)$ and $L(f)$

欠拟合

- Model: assuming we have K trees

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

- Objective

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

Training loss

训练损失函数

Complexity of the Trees

正则项：树的复杂度
一般我们对树的深度，或者数的叶子数量有限制

希望找到一组（K个）树，使得目标函数最小

Loss function 损失函数

- Using Square loss $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$
  - Will results in common gradient boosted machine
- Using Logistic loss $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i)\ln(1 + e^{\hat{y}_i})$
  - Will results in LogitBoost

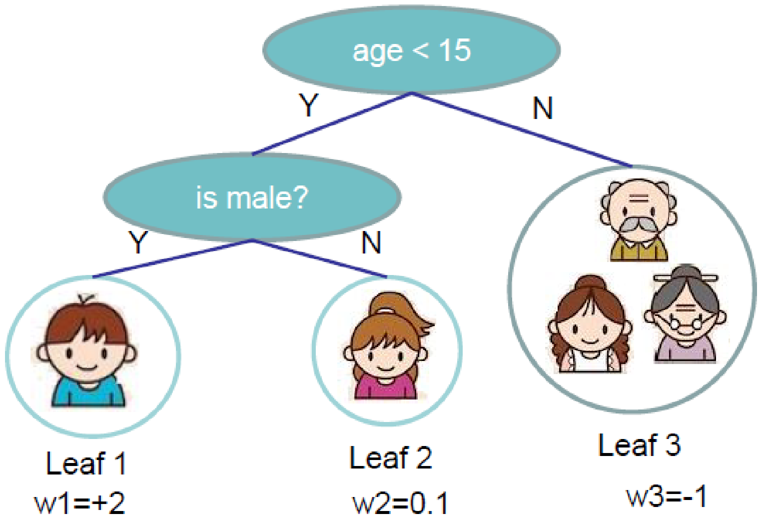
Regularization 正则项：树的复杂度

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

$$\Omega(f_t) = \underbrace{\gamma T}_{\text{Number of leaves}} + \underbrace{\frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2}_{\text{L2 norm of leaf scores}}$$

叶子数量

叶子上的分数的平方和



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

Additive Training

- Solution: **Additive Training (Boosting)**
  - Start from constant prediction, add a new function each time

一轮加一个新函数

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

New function

Model at training round t

第t轮的模型

Keep functions added in previous round

第t轮以前得到的模型

Additive Training

- How to choose f_t 如何确定第t轮的$f_t()$
  - 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

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant
\end{aligned}
$$

Goal: find f_t to minimize this

  - Consider square loss

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} \left(y_i - (\hat{y}_i^{(t-1)} + f_t(x_i))\right)^2 + \Omega(f_t) + const \\
&= \sum_{i=1}^{n} \left[2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2\right] + \Omega(f_t) + const
\end{aligned}
$$

This is usually called residual from previous round 残差项

Additive Training

- 一般情况
  - Goal $Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$
  - Take Taylor expansion of the objective 泰勒展开目标函数
    - Recall $f(x+\Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$
    - Define $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

$$Obj^{(t)} \simeq \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) + constant$$

原来的的Gradient Boosting展开到1阶，xgboost展开到2阶

学习第t个树$f_t()$的目标函数

学习第t个树$f_t()$

- 去掉和$f_t()$无关的项，得到如下目标函数

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

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

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

每个叶子一个二次函数
共T个叶子

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

Simplify the objective 简化目标函数

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

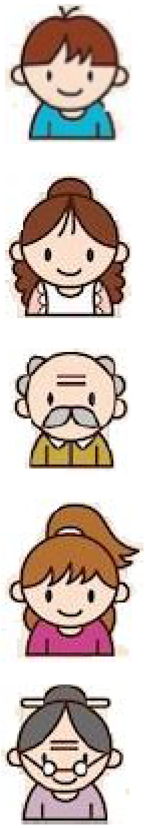
$$
\begin{aligned}
Obj^{(t)} &= \sum_{j=1}^{T}\left[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2\right] + \gamma T \\
&= \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2\right] + \gamma T
\end{aligned}
$$

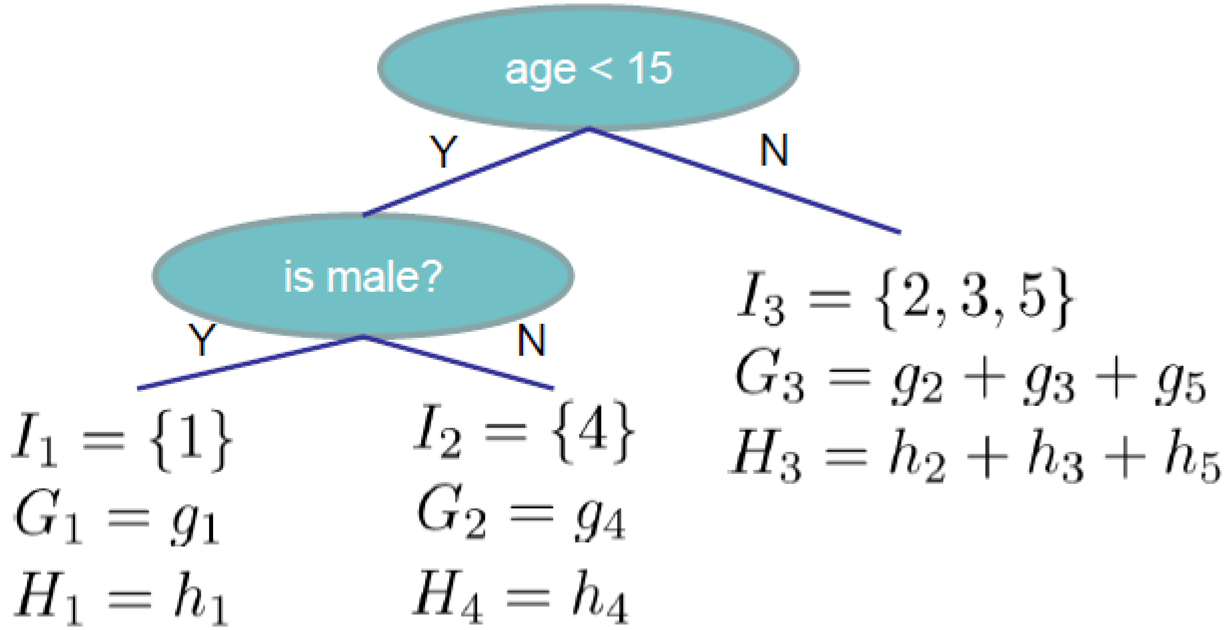
对二次函数，最优的权重取值为：

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

$$argmin_x\ Gx + \frac{1}{2}Hx^2 = -\frac{G}{H},\ H > 0 \qquad \min_x\ Gx + \frac{1}{2}Hx^2 = -\frac{1}{2}\frac{G^2}{H}$$

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





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

The smaller the score is, the better the structure is

Greedily grow the tree 贪心算法建树

- Start with the root 从根节点开始
- Split a leaf 每次对叶子节点进行split
- The gain for a split 一个split对目标函数的贡献

The complexity cost by
introducing additional leaf

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

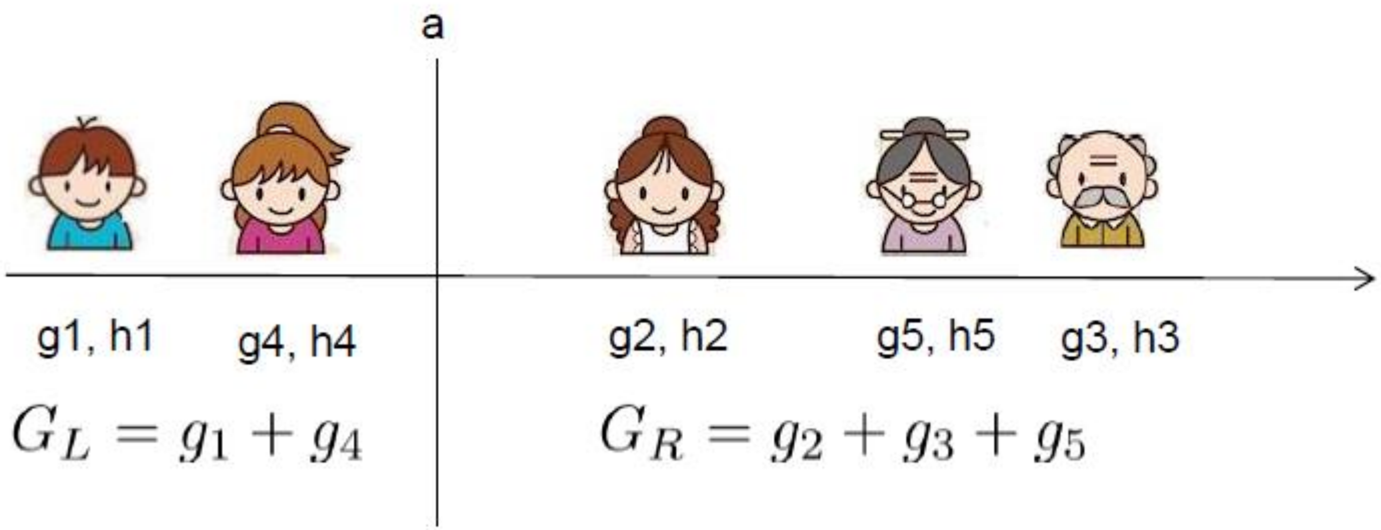
the score of left child

the score of if we do not split

the score of right child

贪心算法建树

- Choose best split 选择最好的split



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

- Stop if gain<0 如果gain是负数就停止

Review 回顾整个算法

- Add a new tree in each iteration　每个循环建立一个树
- Beginning of each iteration, calculate　每个循环初始，计算下列数据

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

- Use the statistics to greedily grow a tree $f_t(x)$　用贪心算法构建树t

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

- Add $f_t(x)$ to the model $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$　将f_t()加入模型
  - Usually, instead we do $y^{(t)} = y^{(t-1)} + \epsilon f_t(x_i)$　实践中会选取一个步长ϵ, 0.1左右，对防止过拟合有帮助
  - ϵ is called step-size or shrinkage, usually set around 0.1
  - This means we do not do full optimization in each step and reserve chance for future rounds, it helps prevent overfitting

Optimization

- Histogram

Too many split points, very expensive

Need to reduce the number of potential split points

(1) Sampling

(2) Use simple quantiles

(3) Xgb used a more involved quantiles

Typically, less than 256 bins are enough

Optimization

Gradient Base One Side Sampling (GOSS) （used in LightGBM）

(analogue to minibatch-SGD)

- a sampling strategy to speedup the training. strategy samples data points according to the magnitudes of gradients before each iteration.

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models $\leftarrow$ {}, fact $\leftarrow \frac{1-a}{b}$
topN $\leftarrow$ a $\times$ len(I) , randN $\leftarrow$ b $\times$ len(I)
for $i = 1$ ***to*** d **do**
  preds $\leftarrow$ models.predict(I)
  g $\leftarrow$ $loss$(I, preds), w $\leftarrow$ {1,1,...}
  sorted $\leftarrow$ GetSortedIndices(abs(g))
  topSet $\leftarrow$ sorted[1:topN]
  randSet $\leftarrow$ RandomPick(sorted[topN:len(I)], randN)
  usedSet $\leftarrow$ topSet + randSet
  w[randSet] $\times=$ fact ▷ Assign weight $fact$ to the small gradient data.
  newModel $\leftarrow$ L(I[usedSet], $-$ g[usedSet], w[usedSet])
  models.append(newModel)

TopN data pts with largest loss (a*100%)

A random subset from remaining data (b*100%)

Combine two subsets

After that, GOSS amplifies the sampled data with small gradients by a constant (1−a)/b when calculating the information gain

Python example 例子

```python
import xgboost as xgb
# read in data
dtrain = xgb.DMatrix('demo/data/agaricus.txt.train')
dtest = xgb.DMatrix('demo/data/agaricus.txt.test')
# specify parameters via map
param = {'max_depth':2, 'eta':1, 'silent':1, 'objective':'binary:logistic' }
num_round = 2
bst = xgb.train(param, dtrain, num_round)
# make prediction
preds = bst.predict(dtest)
```

Data： https://github.com/dmlc/xgboost/tree/master/demo/data

- eta [default=0.3, alias: learning_rate]step size shrinkage used in update to prevents overfitting.
- silent [default=0]0 means printing running messages, 1 means silent mode.
- objective [default=reg:linear]“reg:linear” –linear regression
  - “reg:logistic” –logistic regression
  - “binary:logistic” –logistic regression for binary classification, output probability

参数解释：

- https://xgboost.readthedocs.io/en/latest//parameter.html#general-parameters

数据解释

https://github.com/dmlc/xgboost/blob/master/demo/data/agaricus.txt.train

- 1 3:1 10:1 11:1 21:1 30:1 34:1 36:1 40:1 41:1 53:1 58:1 65:1 69:1 77:1 86:1 88:1 92:1 95:1 102:1 105:1 117:1 124:1 0 3:1 10:1 20:1 21:1 23:1 34:1 36:1 39:1 41:1 53:1 56:1 65:1 69:1 77:1 86:1 88:1 92:1 95:1 102:1 106:1 116:1 120:1
- 0 1:1 10:1 19:1 21:1 24:1 34:1 36:1 39:1 42:1 53:1 56:1 65:1 69:1 77:1 86:1 88:1 92:1 95:1 102:1 106:1 116:1 122:1 1 3:1 9:1 19:1 21:1 30:1 34:1 36:1 40:1 42:1 53:1 58:1 65:1 69:1 77:1 86:1 88:1 92:1 95:1 102:1 105:1 117:1 124:1
- ……

- libsvm的输入数据格式。每行一个数据点。每个数据点包括label和特征，第一个是label，之后的每个含有：的pair是特征的值。：之前表示特征编号，：之后是特征值。(稀疏表示)

备注

- 很多用xgboost的例子 https://github.com/dmlc/xgboost/tree/master/demo#basic-examples-by-tasks
- LightGBM：https://github.com/Microsoft/LightGBM
- Slides主要基于Chen Tianqi的slides修改而成