法律声明

- 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，北风网和讲师拥有完全知识产权；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或者机构不得盗版、复制、仿造其中的创意和内容，我们保留一切通过法律手段追究违反者的权利。
- 课程详情请咨询
  - 微信公众号：北风教育
  - 官方网址：http://www.ibeifeng.com/

人工智能之机器学习

梯度提升树（GBDT）

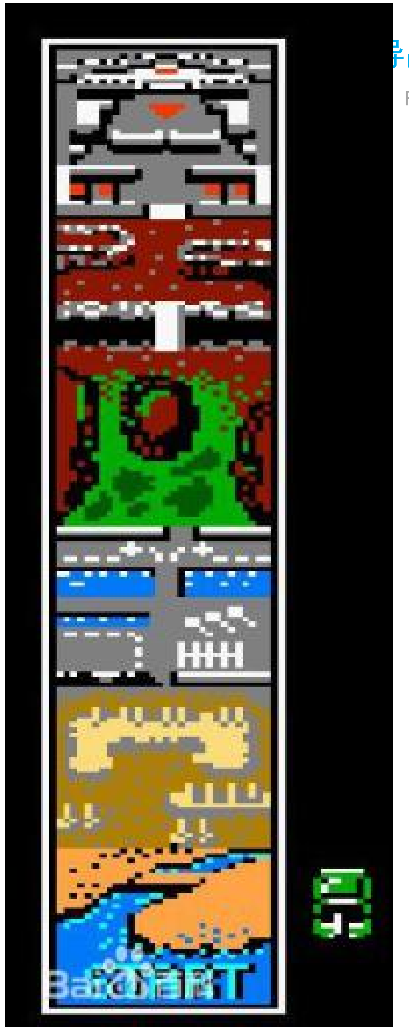
主讲人：赵翌臣

上海育创网络科技有限公司

北风网 IBEIFENG.COM

集成模型一览

- 集成模型
  - boosting模型
    - AdaBoost模型
      - base_estimator KNN
      - ……
      - base_estimator 决策树
    - 提升树
      - GBDT
        - xgboost
        - LightGBM
  - bagging模型
    - 随机森林

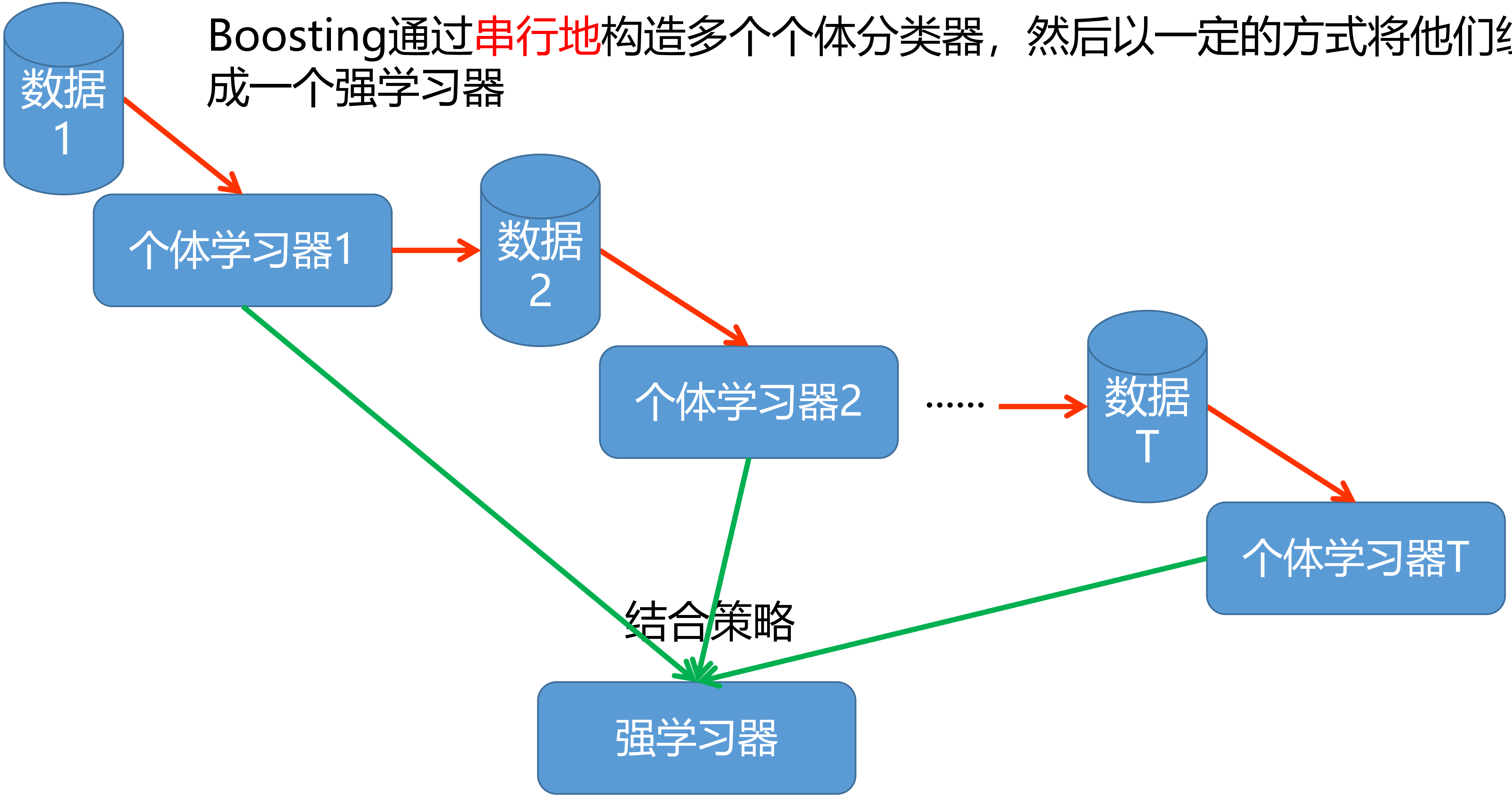


由于是抽象的，也可以叫思想

具体实现

集成学习——Boosting思想

Boosting通过串行地构造多个个体分类器，然后以一定的方式将他们组合成一个强学习器



前向分步学习

介绍

- AdaBoost比较朴素，让后面的学习器更加关注前面学习器预测不好的样本
- 条条大道通罗马，AdaBoost只是建立提升模型的一种方式，还有其他的方式，比如一种更高级的方式——前向分步学习

前向分步学习

- 三个基本元素：加法模型、损失函数、前向分步算法
- AdaBoost的二分类算法是它的一种特例。AdaBoost的二分类算法 ⇔ 前向分步学习的分类算法（模型是加法模型、损失函数为指数损失、学习算法是前向分步算法）
- 还适用于提升树、GBDT、Xgboost、LightGBM等模型的学习

前向分步学习直观解释

round0

初始化加法模型 $f_0(x)=0$

round1

向加法模型加入一个基础学习器，其参数未定，通过构建损失函数，通过最优化求得最优参数

round1

前向分步学习直观解释

round m

round m

……

最终的加法模型

前向分步学习

- 考虑加法模型（additive model）

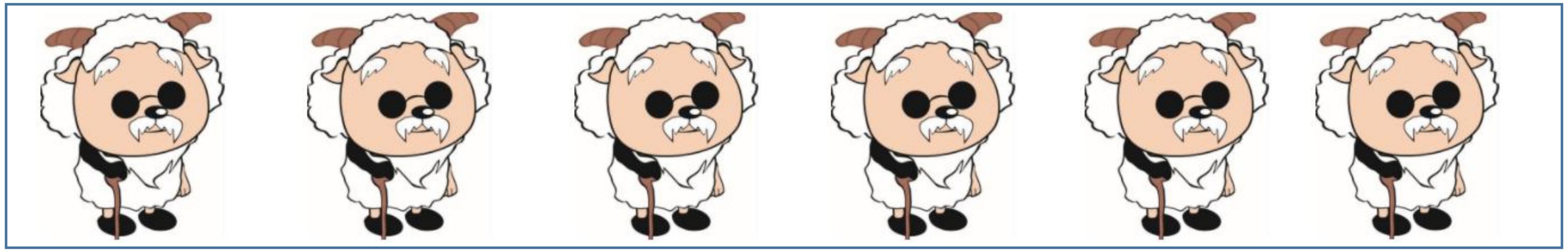
$$f(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$$

- 其中，$b(x;\gamma_m)$为基函数，γ_m为基函数的参数，β_m为基函数的系数。
- 在给定训练数据及损失函数$L(y,f(x))$的条件下，加法模型$f(x)$的学习问题转为代价函数极小化问题：

$$\min_{\{\beta_m,\gamma_m\}_1^M}\sum_{i=1}^{N}L\left(y_i,\sum_{m=1}^{M}\beta_m b(x_i;\gamma_m)\right)$$

- 这是一个复杂的优化问题，可以用前向分步算法（forward stagewise algorithm）求解

前向分步学习



这是一个复杂的优化问题，可以用前向分步算法（forward stagewise algorithm）求解

前向分步学习

- 初始加法模型 $f_0(x)=0$，第m步的加法模型是：

$$f_m(x)=f_{m-1}(x)+\beta_m b(x;\gamma_m)$$

- 其中，$f_{m-1}(x)$ 为当前加法模型，通过经验风险极小化确定基函数的参数

$$(\beta_m,\gamma_m)=\underset{\beta,\gamma}{\arg\min}\sum_{i=1}^{N}L\left(y_i,f_{m-1}(x_i)+\beta b(x_i;\gamma)\right)$$

- 经过M次循环，就得到了加法模型：

$$f(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$$

前向分步算法：将同时求解多个参数的问题转化为逐次求解的问题

前向分步学习的正则化

- 为了防止过拟合，我们通常也会加入学习率(learning rate)。定义为v，对于前面的弱学习器的迭代

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x)$$

如果我们加上了学习率，则有

$$f_m(x) = f_{m-1}(x) + v\beta_m G_m(x)$$

- v的取值范围为0< v ≤1，目的是不让模型一次学的太“到位”，给“后人”留点机会
- 这种结合策略就是之前提过但一直没说的正则后叠加。

提升树（boosting tree）介绍

- 背景
  - 提升树是2000年由Friedman等人提出来的
- 提升树模型可以表示为决策树的加法模型：

$$f(x)=\sum_{m=1}^{M}T(x;\theta_m)$$

- 其中，$T(x;\theta_m)$ 表示决策树；θ_m 为决策树参数；M为树的个数
- 提升树（集成模型），对于分类问题选用二叉分类树（指数损失），对于回归问题选用二叉回归树（平方损失）

提升树（boosting tree）算法

- 提升树算法采用前向分步算法。首先确定初始提升树$f_0(x)=0$，第m步的提升树模型是：

$$f_m(x)=f_{m-1}(x)+T(x;\theta_m)$$

- 其中，$f_{m-1}(x)$为当前加法模型，通过经验风险极小化确定下一棵决策树的参数θ_m

$$\hat{\theta}_m=\underset{\theta_m}{\operatorname{argmin}}\sum_{i=1}^{N}L(y_i,f_{m-1}(x)+T(x;\theta_m))$$

- 经过M次循环，就得到了提升树模型：

$$f(x)=\sum_{m=1}^{M}T(x;\theta_m)$$

提升树做回归任务

- 选择平方损失函数：

$$L(y, f(x)) = (y - f(x))^2$$

- 在加法模型使用上述损失函数

$$L(y, f_{m-1}(x) + T(x;\theta_m)) \\ = (y - f_{m-1}(x) - T(x;\theta_m))^2$$

- 如果令 $r = y - f_{m-1}(x)$，则 r 是当前模型拟合数据的残差（residual）。所以，对于回归问题的提升树算法来说，只需要简单地拟合当前模型的残差。

- 有木有想到之前的一道编程题？

提升树的正则化

- 为了防止提升树过拟合，我们通常也会加入学习率(learning rate)。定义为v，对于前面的弱学习器的迭代

$$f_m(x) = f_{m-1}(x) + T(x;\theta_m)$$

如果我们加上了学习率，则有

$$f_m(x) = f_{m-1}(x) + vT(x;\theta_m)$$

- v的取值范围为0< v ≤1，目的是不让模型一次学的太“到位”，给“后人”留点机会
- 这种结合策略就是之前提过但一直没说的正则后叠加。

梯度提升树（GBDT，2001，Friedman）

背景

- 提升树的学习过程中，当损失函数是平方损失和指数损失时比较好优化，对于其他损失函数不太好优化，即提升树缺少一种通用的优化方案。针对这一问题，freidman提出了梯度提升（Gradient Boosting）
- GBDT的基础学习器为CART树。无论选择什么样的损失函数（使用不同损失函数可以做不同任务），都有一个通用的（general）最优化方案——拟合负梯度!
- 无论做分类还是回归，GBDT拟合的目标都是一个负梯度值（连续值），因此，GBDT的基础学习器只有？？树。

梯度提升树（GBDT）

- GBDT的学习过程（前向分步学习）
  - 三个基本元素：加法模型+损失函数+前向分步算法
  - （1）初始化模型（不用0了）
  - （2）使用拟合负梯度的方式，求得每一轮的基础学习器
  - （3）将基础学习器都叠加（有的可以直接叠加，有的需要做一层转换——调整叶子结点的值），得到集成模型

$$f(x) = \sum_{m=1}^{M} T(x;\theta_m)$$

梯度提升算法

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(y_i, \gamma)$. 初始化加法模型$f_0(x)$，在不同L下，γ取值不同
2. For $m = 1$ to M: 训练M个基础学习器
   (a) For $i = 1, 2, \ldots, N$ compute 对每个样本 i 计算

   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

   损失函数在当前模型下的负梯度

   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$. 将负梯度$r_{im}$作为样本 i 的标签，用回归树拟合，产生$J_m$个叶子结点

   (c) For $j = 1, 2, \ldots, J_m$ compute 对每个叶子结点R_{jm}，计算

   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   不同L下，叶子结点的值不同

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$. 将回归树加入加法模型，这里表示学习率为1
3. Output $\hat{f}(x) = f_M(x)$. 输出最终加法模型

GBDT的损失函数

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

- 均方误差损失函数
  - 损失函数：

$$L(y_i, f(x_i)) = \frac{1}{2} \times (y_i - f(x_i))^2$$

  - 负梯度值为 $y_i - f_{m-1}(x_i)$
- 绝对误差损失函数
  - 损失函数：

$$L(y_i, f(x_i)) = |y_i - f(x_i)|$$

  - 负梯度值为 $sign(y_i - f_{m-1}(x_i))$

GBDT的损失函数

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

log误差损失函数

- 损失函数：

$$L(y_i, f(x_i)) = y_i \log(p_i) + (1 - y_i)\log(1 - p_i), p_i = \frac{1}{1+e^{-f(x_i)}}$$

- 负梯度值为：

$$y_i - \frac{1}{1+e^{-f_{m-1}(x_i)}}$$

对比：对数损失函数不同写法

logistic loss	$\log(1+\exp(-y\mathbf{w}^T\mathbf{x}))$,	$y \in \{-1,+1\}$	$-y\left(1-\frac{1}{1+\exp(-y\mathbf{w}^T\mathbf{x})}\right)\cdot \mathbf{x}$

如果只对f求导，则负梯度变成 $y(\frac{1}{1+e^{yf}})$

-1时：负梯度= $-\frac{1}{1+e^{-f}}$

+1时：负梯度= $\frac{1}{1+e^{f}}$

$$y_i\log(p_i)+(1-y_i)\log(1-p_i), p_i=\frac{1}{1+e^{-f}}, y\in\{0,1\}$$

$$y_i-\frac{1}{1+e^{-f}}$$

0时：负梯度= $-\frac{1}{1+e^{-f}}$

1时：负梯度= $1-\frac{1}{1+e^{-f}}=\frac{1}{1+e^{f}}$

GBDT初始化加法模型

- $f_0(x_i)$初始化为多少？这个取决于loss function：
- 均方误差损失
  - $f_0(x_i) = \bar{y}$ ，$\bar{y}$为样本真实值的平均值
- 绝对误差损失
  - $f_0(x_i) = median_y$ ，用真实值的中位数作为初始值
- log误差损失
  - $f_0(x_i) = \log\left(\frac{\text{正样本个数}}{\text{负样本个数}}\right)$

编程——GBDT回归

- 假设有如下数据集，请根据GBDT的思想，手工实现对数据的建模。提示：需要使用sklearn中的CART回归树作为基础学习器

xi	1	2	3	4	5	6	7	8	9	10
yi	5.56	5.7	5.91	6.4	6.8	7.05	8.9	8.7	9	9.05

- 初始化，作为加法模型的初始 $f_0(x_i) = \bar{y} = 7.306$

- 均方误差损失函数：$L(y_i, f(x_i)) = \frac{1}{2} \times (y_i - f(x_i))^2$，当前加法模型下损失函数负梯度值为 $y_i - f_{m-1}(x_i)$，由公式，可以计算负梯度值：

xi	1	2	3	4	5	6	7	8	9	10
yi	-1.747	-1.607	-1.397	-0.907	-0.507	-0.257	1.593	1.393	1.693	1.743

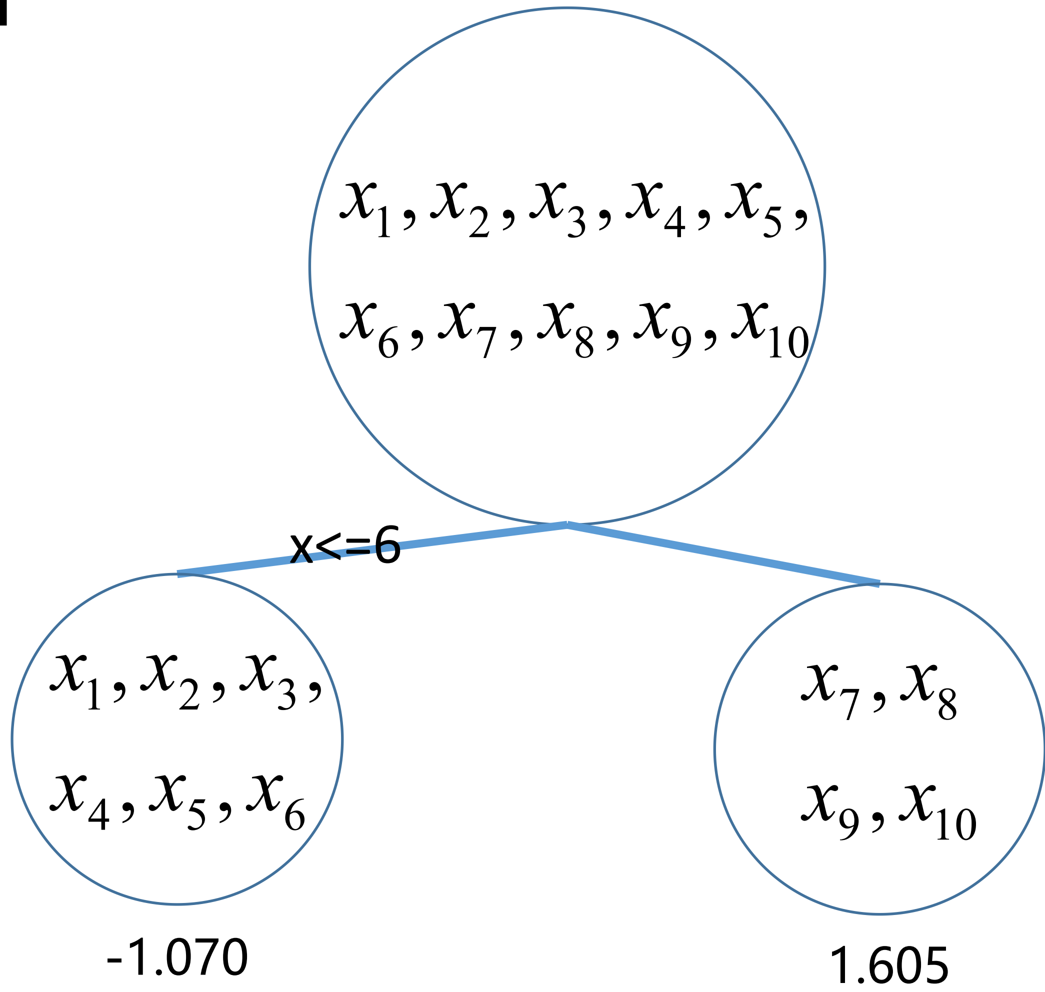
编程——GBDT回归

- 下面就是以负梯度值作为目标进行拟合，使用CART回归树，建树算法略，得到如下回归树，再将回归树加入到加法模型

$$f_m(x) = f_{m-1}(x) + 0.1 * h(x\,;\alpha_m)$$

- 每个样本的y都向着正确的方向开始靠拢
- 比如y_1=5.56，初始加法模型预测为7.307，加入第一棵回归树后变为

$$f_m(x_1) = 7.307 - 0.1*1.070 = 7.199$$



编程——GBDT回归

- 再根据 $y_i - f_{m-1}(x_i)$计算负梯度值，比如：5.556-7.199=-1.64

xi	1	2	3	4	5	6	7	8	9	10
yi	-1.64	-1.50	-1.29	-0.80	-0.40	-0.15	1.43	1.23	1.53	1.58

- 再以该负梯度值进行拟合，构建回归树
- 循环执行，直到得到M个回归树的GBDT模型

编程——GBDT分类

- 假设有如下数据集，请根据GBDT的思想，手工实现对数据的建模。提示：需要使用sklearn中的回归树作为基础学习器

xi	1	2	3	4	5	6	7	8	9	10
yi	0	0	0	1	1	0	0	0	1	1

- 初始化，作为加法模型的初始 $f_0(x_i) = \log\left(\frac{\text{正样本个数}}{\text{负样本个数}}\right) = \log(\frac{4}{6}) = -0.4054$

- log误差损失函数：

$$L(y_i, f(x_i)) = y_i \log(p_i) + (1 - y_i)\log(1 - p_i) \;\; , p_i = \frac{1}{1 + e^{-f(x_i)}}$$

编程——GBDT分类

- 那么其损失函数负梯度值为：

$$y_i - \frac{1}{1 + e^{-f_{m-1}(x_i)}}$$

- 由公式，可以计算负梯度值，以x1和x4为例

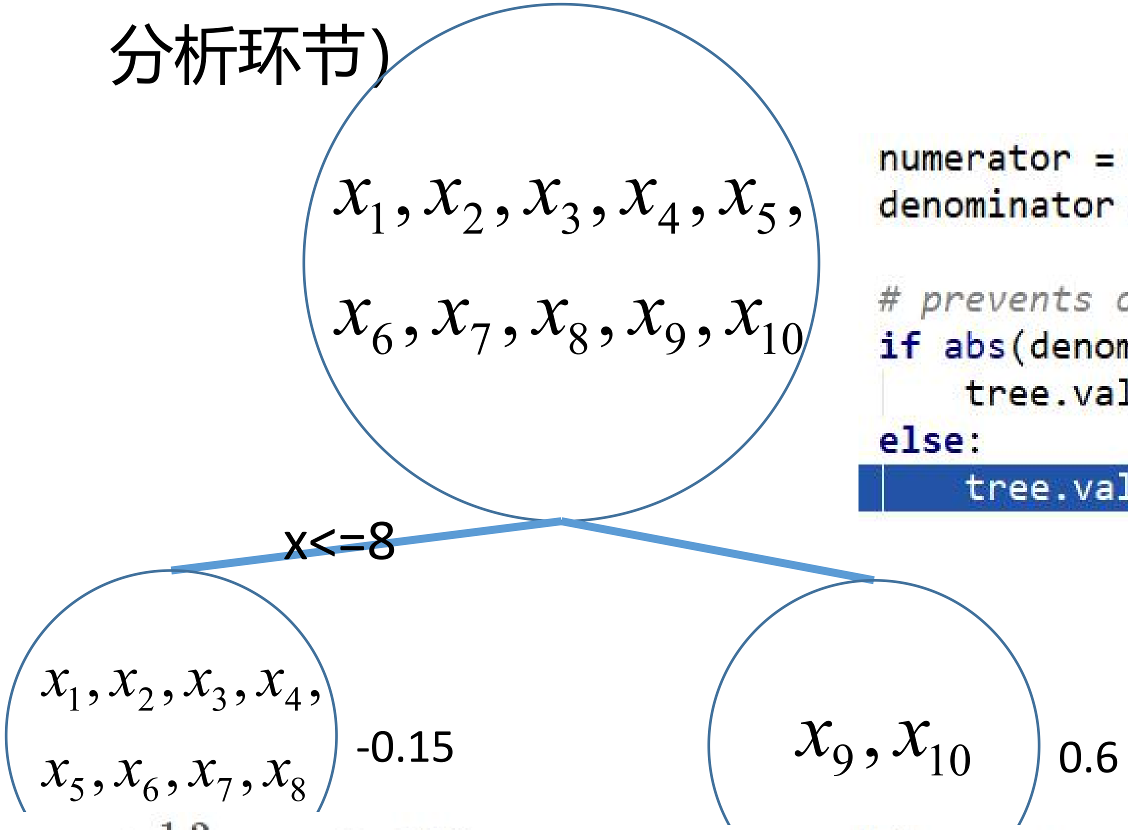
$$\tilde{y}_1 = 0 - \frac{1}{1 + e^{0.4054}} = -0.4, \qquad \tilde{y}_4 = 1 - \frac{1}{1 + e^{0.4054}} = 0.6$$

- 得到

xi	1	2	3	4	5	6	7	8	9	10
yi	-0.4	-0.4	-0.4	0.6	0.6	-0.4	-0.4	-0.4	0.6	0.6

编程——GBDT分类

- 下面就是以负梯度值作为目标进行拟合，使用CART回归树，建树算法略，得到如下回归树，对叶子结点值进行调整，再将回归树加入到加法模型（详情见源码分析环节）




```
numerator = np.sum(sample_weight * residual)  numerator: -1.2000000000000
denominator = np.sum(sample_weight * (y - residual) * (1 - y + residual))
                                                 + {ndarray} [-0.4 -0.4 -0.4  0.6  0.6 -0.4 -0.4 -0.4]

# prevents overflow and division by zero
if abs(denominator) < 1e-150:
    tree.value[leaf, 0, 0] = 0.0
else:
    tree.value[leaf, 0, 0] = numerator / denominator
                                        {float64} 1.92
```

$$\gamma_{11} = \frac{-1.2}{1.92} = -0.625 \qquad \gamma_{21} = \frac{1.2}{0.480} = 2.5$$

编程——GBDT分类

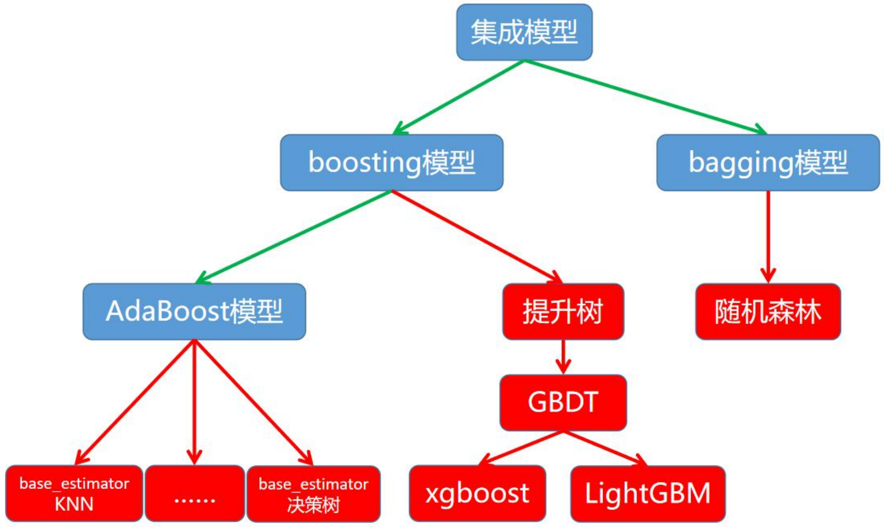
- 再根据 $y_i - \frac{1}{1+e^{-f_{m-1}(x_i)}}$ 计算负梯度值
- 再以该负梯度值进行拟合，构建回归树，叶子结点调整，加入加法模型
- 循环执行，直到得到M个回归树的GBDT模型

GBDT小结

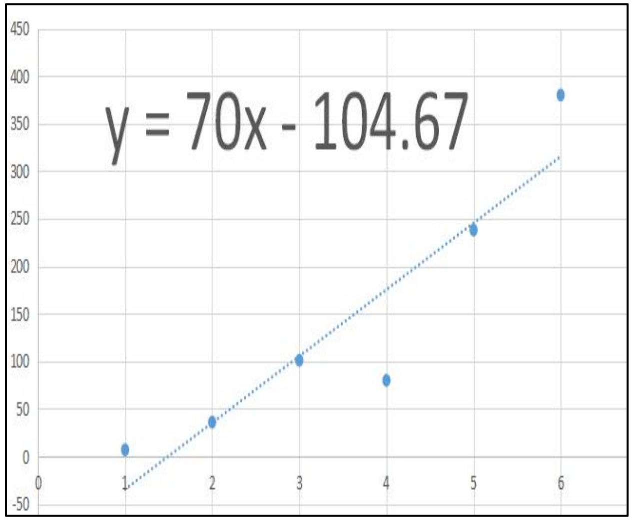
- GBDT也是可以设置学习率的
- 优点
  - GBDT能处理类别特征，能扩展到多类分类，不需要特征缩放，并且能够捕获特征间的非线性关系和特征的相互影响
  - 在相对少的调参时间情况下，预测的准确率也可以比较高。
- 缺点
  - 由于弱学习器之间存在依赖关系，难以并行训练数据。
  - 有过拟合的风险

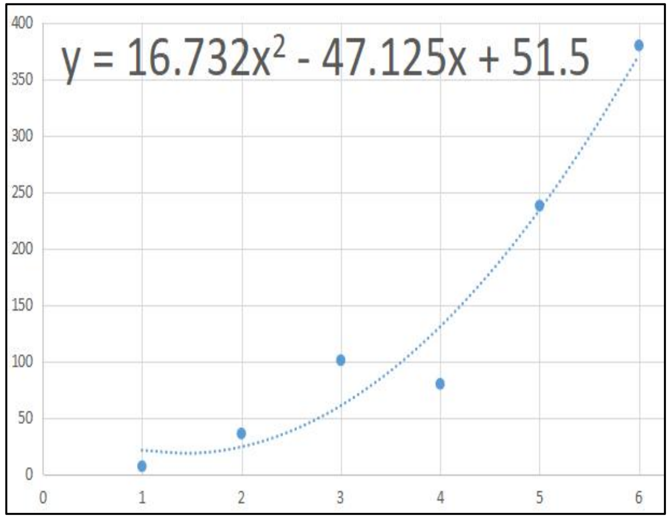
随机森林 VS 梯度提升树

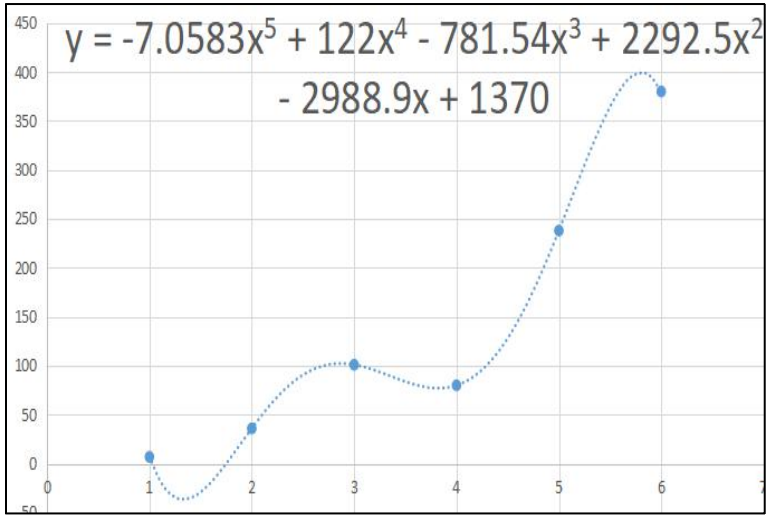
- GBDT是串行的算法；随机森林是并行的算法
- GBDT训练的决策树深度比较浅，随机森林比较深
- GBDT容易过拟合（高方差），随机森林不容易过拟合
- GBDT需要调树的个数，随机森林不需要调树的个数



模型的偏差与方差







高偏差（high bias）　　just right　　高方差（high variance）

欠拟合（underfitting）　　过拟合（overfitting）

编程——基于Boosting的回归

例 8.2 已知如表 8.2 所示的训练数据，x 的取值范围为区间[0.5,10.5]，y 的取值范围为区间[5.0,10.0]，学习这个回归问题的提升树模型，考虑只用树桩作为基函数。

表 8.2 训练数据表

x_i	1	2	3	4	5	6	7	8	9	10
y_i	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05

并行的训练多颗回归树，对有个样本进行预测时，所有回归树同时预测，取均值作为输出

编程——基于Boosting的分类

例 8.1 给定如表 8.1 所示训练数据．假设弱分类器由 $x<v$ 或 $x>v$ 产生，其阈值 v 使该分类器在训练数据集上分类误差率最低．试用 AdaBoost 算法学习一个强分类器．

表 8.1 训练数据表

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	−1	−1	−1	1	1	1	−1

编程——GBDT综合案例之森林植被类型预测


- 数据集：
  - https://archive.ics.uci.edu/ml/datasets/covertype
- 解释：
  - 该数据集记录了美国科罗拉多州不同地块的森林植被类型。每个样本包含了描述每块土地的若干特征，包括海拔、坡度、到水源的距离、遮阳情况和土壤类型，并且随同给出了地块的已知森林植被类型。我们需要总共54 个特征中的其余各项来预测森林植被类型

Data Set Characteristics:	Multivariate	Number of Instances:	581012	Area:	Life
Attribute Characteristics:	Categorical, Integer	Number of Attributes:	54	Date Donated	1998-08-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	185453

编程——GBDT回归案例之共享单车租赁数量预测

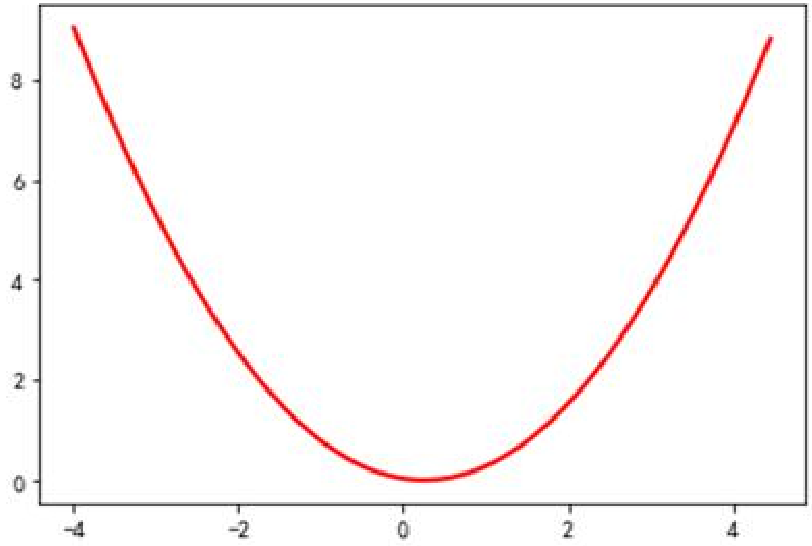
- This dataset contains the hourly and daily count of rental bikes between years 2011 and 2012 in Capital bikeshare system with the corresponding weather and seasonal information.
  - 数据下载 http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

Data Set Characteristics:	Univariate	Number of Instances:	17389	Area:	Social
Attribute Characteristics:	Integer, Real	Number of Attributes:	16	Date Donated	2013-12-20
Associated Tasks:	Regression	Missing Values?	N/A	Number of Web Hits:	232895

思考：为什么拟合负梯度

假设某样本的标签=0.25

损失函数 $Loss(y_i, f(x_i)) = 0.5 * (f(x_i) - 0.25)^2$



损失函数的负梯度为：$0.25 - f(x_i)$

模型预测值 $f(x_i)$

当预测值为4时，损失值=0.5*(4-0.25)²=7.03

当前负梯度为-3.75，用新的决策树去拟合负梯度，然后将该树加到加法模型。比如新的决策树会预测这个样本为-3，如果学习率=0.1，那么加法模型会将这个样本预测为4-0.3=3.7，从而降低了损失。这个逻辑适用于全部样本，因此，通过拟合负梯度产生决策树，将决策树加入加法模型，能降低代价函数

IBEIFENG.COM

THANK YOU

上海育创网络科技有限公司

PS：平方误差直接加！

4.1 Least–squares regression

Here $L(y, F) = (y - F)^2/2$. The pseudo–response in line 3 of Algorithm 1 is $\tilde{y}_i = y_i - F_{m-1}(\mathbf{x}_i)$. Thus, line 4 simply fits the current residuals and the line search (line 5) produces the result $\rho_m = \beta_m$, where β_m is the minimizing β of line 4. Therefore, gradient boosting on squared–error loss produces the usual *stagewise* approach of iteratively fitting the current residuals:

Algorithm 2: LS_Boost

$F_0(\mathbf{x}) = \bar{y}$

For $m = 1$ to M do:

$\quad \tilde{y}_i = y_i - F_{m-1}(\mathbf{x}_i), \quad i = 1, N$

$\quad (\rho_m, \mathbf{a}_m) = \arg\min_{\mathbf{a},\rho} \sum_{i=1}^{N} [\tilde{y}_i - \rho h(\mathbf{x}_i; \mathbf{a})]^2$

$\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

endFor

end Algorithm

PS：绝对误差

4.2 Least-absolute-deviation (LAD) regression

For the loss function $L(y, F) = |y - F|$, one has

$$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} = sign(y_i - F_{m-1}(\mathbf{x}_i)).$$

Algorithm 3: LAD_TreeBoost

$F_0(\mathbf{x}) = median\{y_i\}_1^N$

For $m = 1$ to M do:

  $\tilde{y}_i = sign(y_i - F_{m-1}(\mathbf{x}_i)),\ i = 1, N$

  $\{R_{jm}\}_1^J = J$–terminal node $tree(\{\tilde{y}_i, \mathbf{x}_i\}_1^N)$

  $\gamma_{jm} = median_{\mathbf{x}_i \in R_{jm}}\{y_i - F_{m-1}(\mathbf{x}_i)\},\ j = 1, J$

  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \sum_{j=1}^J \gamma_{jm} 1(\mathbf{x} \in R_{jm})$

endFor

end Algorithm

PS：Huber误差

4.4 M–Regression

M–regression techniques attempt resistance to long–tailed error distributions and outliers while maintaining high efficiency for normally distributed errors. We consider the Huber loss function (Huber 1964)

$$L(y,F) = \begin{cases} \frac{1}{2}(y-F)^2 & |y-F| \le \delta \\ \delta(|y-F| - \delta/2) & |y-F| > \delta \end{cases}. \tag{19}$$

Here the pseudo–response is

$$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} = \begin{cases} y_i - F_{m-1}(\mathbf{x}_i) & |y_i - F_{m-1}(\mathbf{x}_i)| \le \delta \\ \delta \cdot sign(y_i - F_{m-1}(\mathbf{x}_i)) & |y_i - F_{m-1}(\mathbf{x}_i)| > \delta \end{cases}.$$

Algorithm 4: M_TreeBoost

$F_0(\mathbf{x}) = median\{y_i\}_1^N$

For $m = 1$ to M do:

$r_{m-1}(\mathbf{x}_i) = y_i - F_{m-1}(\mathbf{x}_i),\ i = 1, N$

$\delta_m = \text{quantile}_\alpha\{|r_{m-1}(\mathbf{x}_i)|\}_1^N$

$$\tilde{y}_i = \begin{cases} r_{m-1}(\mathbf{x}_i) & |r_{m-1}(\mathbf{x}_i)| \le \delta_m \\ \delta_m \cdot sign(r_{m-1}(\mathbf{x}_i)) & |r_{m-1}(\mathbf{x}_i)| > \delta_m \end{cases},\ i = 1, N$$

$\{R_{jm}\}_1^J = J$–terminal node $tree(\{\tilde{y}_i, \mathbf{x}_i\}_1^N)$

$\tilde{r}_{jm} = median_{\mathbf{x}_i \in R_{jm}} \{r_{m-1}(\mathbf{x}_i)\},\ j = 1, J$

$\gamma_{jm} = \tilde{r}_{jm} + \frac{1}{N_{jm}} \sum_{\mathbf{x}_i \in R_{jm}} sign(r_{m-1}(\mathbf{x}_i) - \tilde{r}_{jm}) \cdot \min(\delta_m, abs(r_{m-1}(\mathbf{x}_i) - \tilde{r}_{jm})),$

$j = 1, J$

$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \sum_{j=1}^J \gamma_{jm} 1(\mathbf{x} \in R_{jm})$

endFor

end Algorithm

PS：对数误差

4.5 Two–class logistic regression and classification

Here the loss function is negative binomial log–likelihood (FHT00)

$$L(y, F) = \log\left(1 + \exp(-2yF)\right), \quad y \in \{-1, 1\},$$

where

$$F(\mathbf{x}) = \frac{1}{2}\log\left[\frac{\Pr(y=1\,|\,\mathbf{x})}{\Pr(y=-1\,|\,\mathbf{x})}\right].$$

The pseudo–response is

$$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} = 2y_i/(1+\exp(2y_iF_{m-1}(\mathbf{x}_i))).$$

Algorithm 5: $\mathbf{L_2}$_TreeBoost

$F_0(\mathbf{x}) = \frac{1}{2}\log\frac{1+\bar{y}}{1-\bar{y}}$

For $m = 1$ to M do:

  $\tilde{y}_i = 2y_i/(1+\exp(2y_iF_{m-1}(\mathbf{x}_i))),\ i = 1, N$

  $\{R_{jm}\}_1^J = J$–terminal node $tree(\{\tilde{y}_i, \mathbf{x}_i\}_1^N)$

  $\gamma_{jm} = \sum_{\mathbf{x}_i\in R_{jm}} \tilde{y}_i / \sum_{\mathbf{x}_i\in R_{jm}} |\tilde{y}_i|\,(2-|\tilde{y}_i|),\ j = 1, J$

  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \sum_{j=1}^J \gamma_{jm} 1(\mathbf{x}\in R_{jm})$

endFor

end Algorithm