第7讲
决策树与集成学习
Decision Tree and Ensemble Learning

决策树算法：ID3和C4.5

分类和回归树：CART

Bagging和随机森林

提升、AdaBoost和提升树

1. 决策树

- 决策树（decision tree）是一种分层的决策结构，可用于分类和回归；
- 决策树对于特征向量各分量混合属性的情形，尤其有效
- 决策树模型具有树型结构，学习过程中由样本集形成一棵可做分层判决的树；
- 推断（预测）时，对于一个新的特征向量，从树的根结点起分层判决，达到可给出最后结果的叶结点，完成一次推断；
- 决策树推断速度快，可解释性强，是一种应用非常广泛的算法；
- 以决策树为基础树，可直接形成随机森林或提升树（集成学习），得到更高的分类率或回归精度。

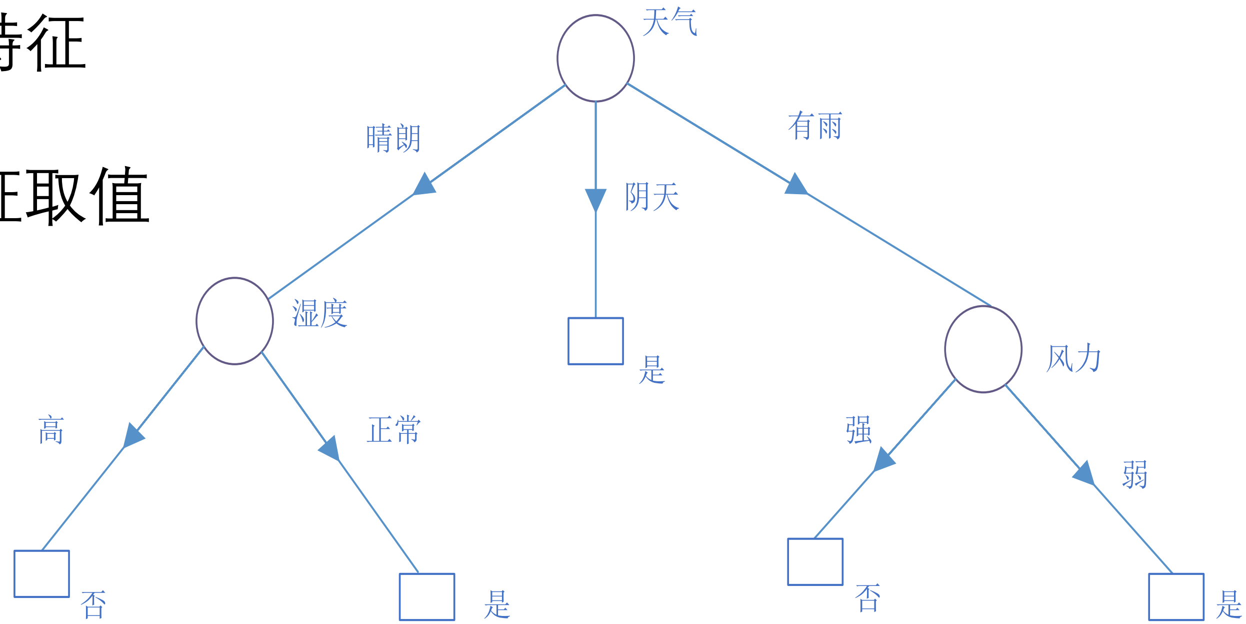
样本集

决策树的一些术语

$$\mathbf{D} = \left\{ \left(\boldsymbol{x}_n, y_n \right) \right\}_{n=1}^{N}$$

决策树的结构是分层的结构，由结点和边组成

特征向量有若干特征
可离散、连续
在一个结点，特征取值决定边的分支



根结点

内部结点

叶结点：
可以得出结果的结点

2. ID3算法和信息增益

ID3算法基本思路：（每个特征取值离散）

- 通过一个准则找到一个最合适的特征，以这个特征作为根结点；
- 根结点对应了样本集的所有样本，确定一个特征作为根结点，由该特征的各取值形成相应的几条边，引出下一层的结点；
- 每一条边对应了根结点特征的一个取值，根据该特征的不同取值把样本集分成几组，每一组形成一子样本集；
- 子样本集的特征向量中，将根结点已用到的特征删去，每一个子样本集对应一条边下端的结点；
- 满足条件的节点定为叶节点，并确定对应输出，否则递归执行。

不纯性度量（impurity measure）

ID3算法时用熵表示不纯性

$$H(\mathbf{D}) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

实际计算经验熵

第 k 类的样本数目为 c_k，则 p_k 估计为 $p_k \approx \frac{c_k}{N}$

$$H(\mathbf{D}) = -\sum_{k=1}^{K} \frac{c_k}{N} \log_2 \frac{c_k}{N}$$

不纯性度量（续）

考虑了特征 A 后的不纯性可用条件熵表示，即 $H\left(\mathbf{D}\middle|A\right)$

$$H\left(\mathbf{D}\middle|A\right)=\sum_{i\in[1,I]}p_i^{(A)}H\left(\mathbf{D}\middle|A=i\right)$$

$p_i^{(A)}$ 表示特征 $A=i$ 的概率

$$H\left(\mathbf{D}\middle|A\right)=-\sum_{i\in[1,I]}\frac{N_i}{N}\sum_{k=1}^{K}\frac{c_{ik}}{N_i}\log_2\frac{c_{ik}}{N_i}$$

不纯性度量（续）

定义选择了特征 A 的熵增益为

$$G(\mathbf{D}, A) = H(\mathbf{D}) - H(\mathbf{D}|A)$$

对每个特征 A，计算熵增益

选择熵增益最大的特征作为根结点对应的特征

由特征的取值形成边，边的下端为下一层的结点

各自对应了子样本集 $\mathbf{D}_i$

ID3算法

ID3（$\mathbf{D}$，分类类型，特征）

（1）　创建树的根结点

若 $\mathbf{D}$ 中的所有标注都相同，返回一个单一根结点树，输出为该标注类

若特征向量为空，返回一个单一根结点树，输出为标注最多的类

（2）　计算各特征的信息增益，将信息增益最大的特征记为 A

A 作为根结点特征

对于 A 的每一个取值 i

在根结点下加一条新的边，其对应测试条件为 $A=i$

令 $\mathbf{D}_i$ 为 $\mathbf{D}$ 中满足 $A=i$ 的样本子集

如果 $\mathbf{D}_i$ 为空，

该边下端设为叶结点，叶结点的输出为 $\mathbf{D}$ 中标注最多的类

否则，

在这个边下端加一个新的子树：调用 ID3（$\mathbf{D}_i$，分类类型，特征-$\{A\}$）

（3）返回根

例：看电影样本集

序号	女朋友 A_1	作业 A_2	预习 A_3	电影类型 A_4	决定
1	女友去	完成	需要	喜欢	看电影
2	女友去	未完成	需要	不喜欢	不去
3	女友去	未完成	不需要	不喜欢	看电影
4	女友去	完成	需要	不喜欢	看电影
5	女友不去	完成	不需要	喜欢	看电影
6	女友不去	未完成	不需要	喜欢	不去
7	女友不去	完成	需要	喜欢	看电影
8	女友不去	完成	不需要	不喜欢	不去
9	女友不去	未完成	需要	不喜欢	不去
10	无女友	完成	不需要	喜欢	看电影
11	无女友	未完成	不需要	喜欢	不去
12	无女友	未完成	需要	喜欢	不去
13	无女友	完成	不需要	不喜欢	不去
14	无女友	未完成	不需要	喜欢	不去
15	无女友	完成	需要	喜欢	看电影

数据集的经验熵

$$H(\mathbf{D})=-\frac{7}{15}\log_2\frac{7}{15}-\frac{8}{15}\log_2\frac{8}{15}=0.9966$$

每个特征的增益

$$G(\mathbf{D},A_1)=H(\mathbf{D})-H(\mathbf{D}|A_1)$$

$$=0.9966-\frac{4}{15}\left(-\frac{3}{4}\log_2\frac{3}{4}-\frac{1}{4}\log_2\frac{1}{4}\right)-\frac{5}{15}\left(-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}\right)$$

$$-\frac{6}{15}\left(-\frac{2}{6}\log_2\frac{2}{6}-\frac{4}{6}\log_2\frac{4}{6}\right)=0.0866$$

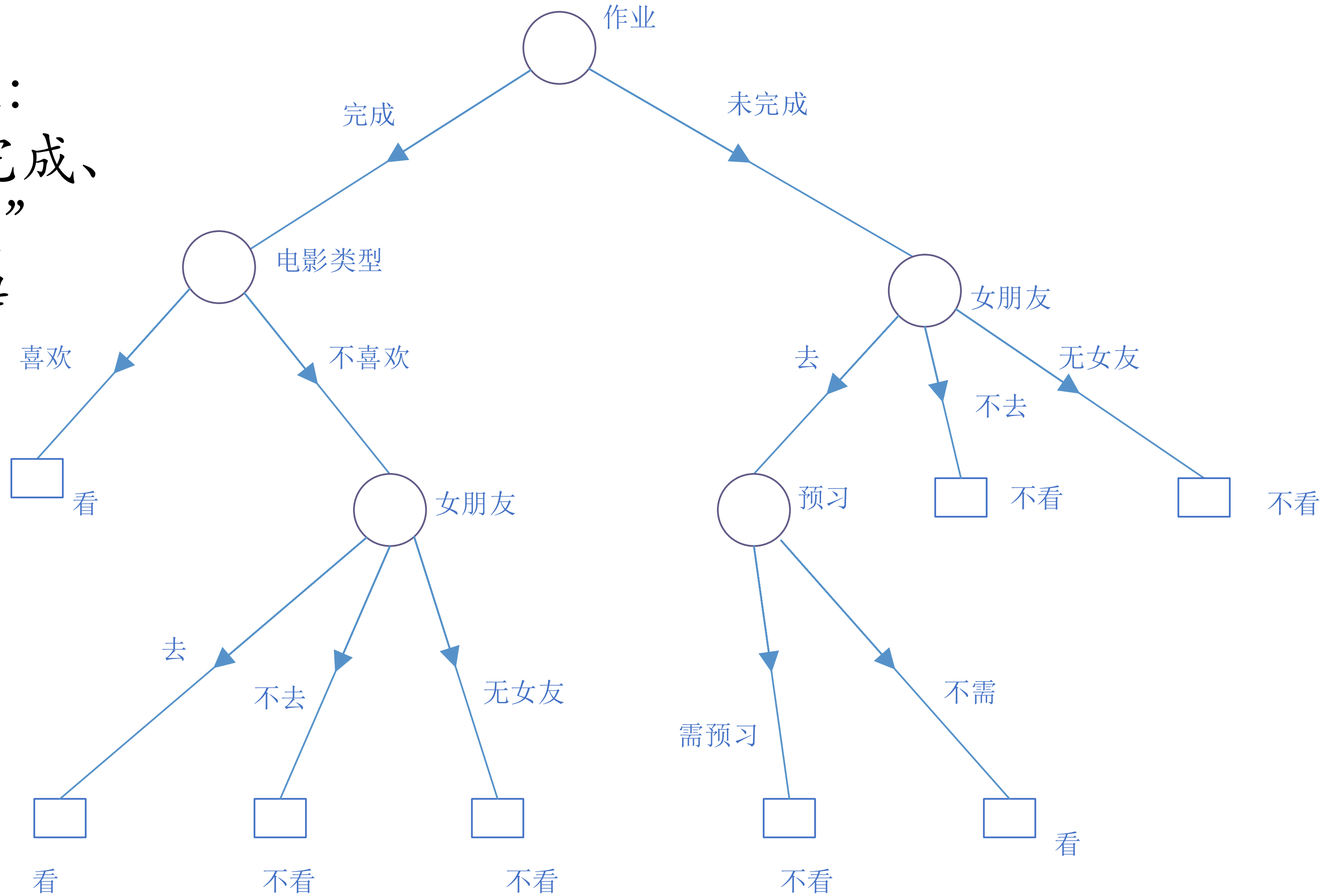
$$G(\mathbf{D},A_2)=0.2876$$

$$G(\mathbf{D},A_3)=0.027$$

$$G(\mathbf{D},A_4)=0.0366$$

最终的决策树

一个新的输入：
女友去、未完成、
不需要、喜欢”
最后的决策是
“去看电影”



3. C4.5算法

- Quinian于1993年发表;
- ID3中用信息增益选择特征时偏向选择取值数多的特征，C4.5用信息增益率来选择特征，适用于特征之间取值数目比较分散的情况;
- 针对ID3易于过拟合的问题，在树构造过程中引入剪枝技术;
- ID3的特征只能取离散值，若一个特征是连续数值量，则需要预先离散化，C4.5既可以处理离散特征，也可处理连续特征;
- 能够处理样本特征不完整的情况。

CART算法

- CART是分类与回归树的缩写：Classification And Regression Tree: CART。由Breiman等1984年提出。
- CART的结构一般是一棵二叉树，每一个内部节点有两条分支
- CART用于分类，称为分类树；用于回归，称为回归树。

4.1 分类树

考虑二叉树结构

假设特征 A 的取值可为 $1,2,\cdots,I$

分别判断 $A=i$ 时，对分类不纯性的改善

当 $A=j$ 时，对不纯性的改善最佳

则取 $A=j$ 为该结点的判据

将分为 $A=j$ 和 $A\neq j$ 的两个分支

CART算法的不纯性度量：基尼指数

对于样本集表示为 $\mathbf{D}$

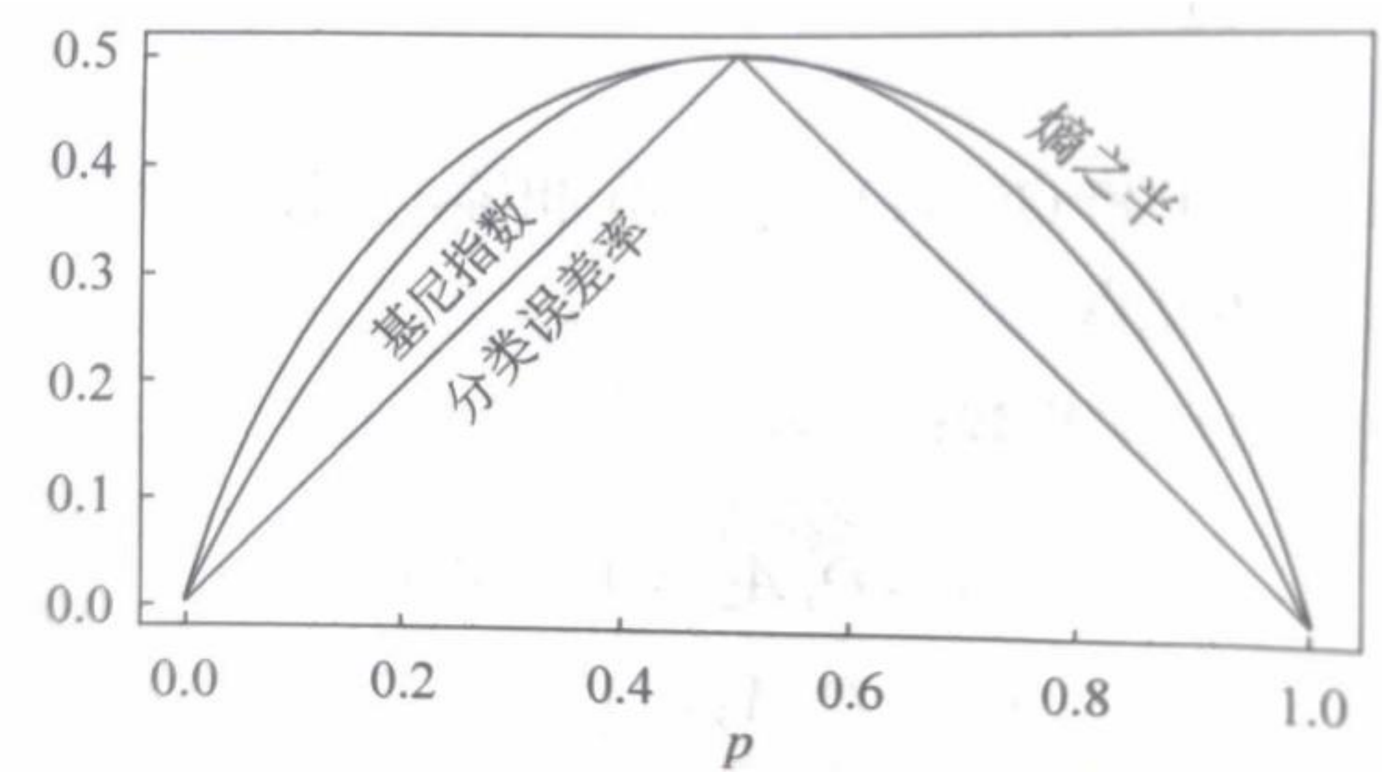
$$Gini(\mathbf{D}) = \sum_{k=1}^{K}\sum_{j\neq k} p_k p_j$$

$$= \sum_{k=1}^{K} p_k(1-p_k) = 1-\sum_{k=1}^{K} p_k^2$$

若 p_k 估计为 $p_k \approx \frac{c_k}{N}$

基尼指数计算为 $$Gini(\mathbf{D}) = 1-\sum_{k=1}^{K}\left(\frac{c_k}{N}\right)^2$$

两类情况下，熵的一半与基尼指数的关系图



基尼指数作为测试条件

当测试特征 $A=i$ 时，

按照样本的特征 A 是否满足 $A=i$ 将样本集分成两类

$$\mathbf{D}_1=\left\{\left(\boldsymbol{x}_n,y_n\right)\middle|A\left(\boldsymbol{x}_n\right)=i\right\},\ \mathbf{D}_2=\left\{\left(\boldsymbol{x}_n,y_n\right)\middle|A\left(\boldsymbol{x}_n\right)\neq i\right\}$$

在特征 $A=i$ 的条件下，基尼指数为

$$Gini\left(\mathbf{D},A=i\right)=\frac{N_1}{N}Gini\left(\mathbf{D}_1\right)+\frac{N_2}{N}Gini\left(\mathbf{D}_2\right)$$

设 $Gini\left(\mathbf{D},A=j\right)$ 最小，则选择特征 A，以 $A=j$ 是或否为分支

CART-C（$\mathbf{D}$，分类类型，特征）

（1） 创建树的根结点

若 $\mathbf{D}$ 中的所有标注都相同，返回一个单一根结点树，输出为该标注类

若特征向量为空，返回一个单一根结点树，输出为标注最多的类

若一个特征 A 只有一个取值，删除该特征

（2） 计算各特征和特征的各取值的 Gini 指数（若特征 A 只有两个取值，只计算第一个值的基尼指数，具有最小基尼指数的特征和取值为 $A=j$，以是或否 $A=j$，将样本集分为 $\mathbf{D}_1,\mathbf{D}_2$，分为两条支路：

$A=j$ "是"支路，调用 CART-C（$\mathbf{D}_1$，分类类型，特征-$\{A\}$）

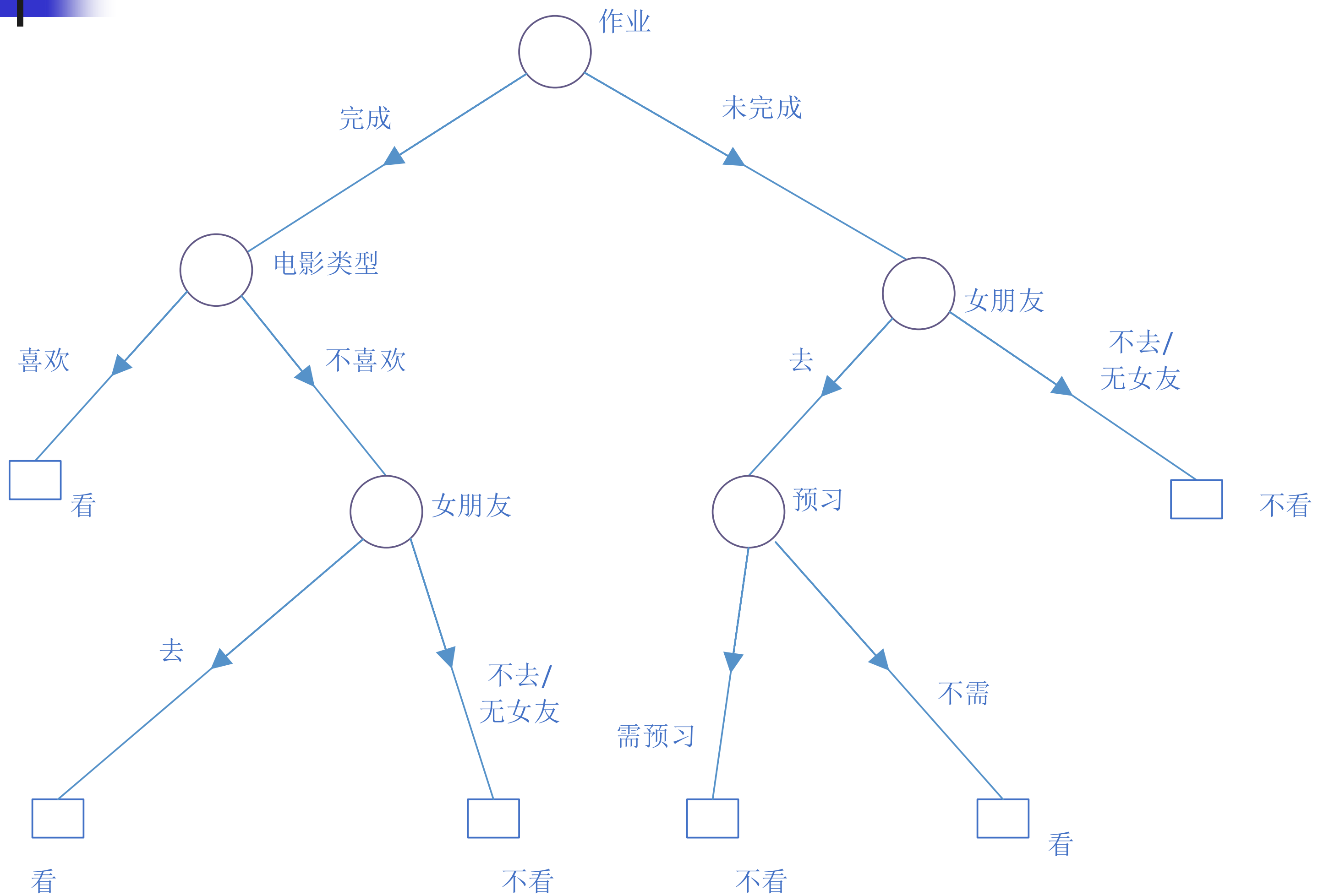
$A=j$ "否"支路，调用 CART-C（$\mathbf{D}_2$，分类类型，特征 A 取值集合-$\{j\}$）

（3） 若 $\mathbf{D}_i$ 为空，结点设为叶结点，叶结点的输出为 $\mathbf{D}$ 中标注最多的类

若 $\mathbf{D}_i$ 只有一种标注，结点设为叶结点，输出为该标注

若特征为空，结点设为叶结点，$\mathbf{D}_i$ 中最多类型的标注为输出

看电影的CART分类树



4.2 CART回归树

回归树的不纯性函数使用平方误差

在一个样本集 $\mathbf{D}$ 中 x_i 出现的所有取值的集合记为 A_i

对每一个特征 A_i 的每一个取值 $a \in A_i$

将特征空间划分为两个区域

$$\mathbf{R}_1(A_i,a)=\{\boldsymbol{x}|x_i \le a\},\quad \mathbf{R}_2(A_i,a)=\{\boldsymbol{x}|x_i > a\}$$

$a \in A_i$ 称为切分点

回归树：

对应的样本集分为两个子样本集

$$\mathbf{D}_1=\{(\boldsymbol{x}_n,y_n)|x_{ni}\le a,n\in[1,N]\}$$

$$\mathbf{D}_2=\{(\boldsymbol{x}_n,y_n)|x_{ni}> a,n\in[1,N]\}$$

对每一个子样本集的样本，

用一个常数 $\hat{g}_m$, $m=1,2$ 逼近，则逼近误差为

$$\sum_{m=1}^{2}\sum_{(\boldsymbol{x}_n,y_n)\in\mathbf{D}_m}(y_n-\hat{g}_m)^2$$

求 $\hat{g}_m$, $m=1,2$ 使得平方误差和最小

最优值 $$\hat{g}_m=\frac{1}{N_m}\sum_{(\boldsymbol{x}_n,y_n)\in\mathbf{D}_m}y_n,\qquad m=1,2$$

回归树：

在根结点需要选出一个特征和一个切分点，使得

$$\left(A_j, a_o\right) = \underset{A_i, a \in A_i}{\arg\min} \left\{ \sum_{m=1}^{2} \sum_{(\boldsymbol{x}_n, y_n) \in \mathbf{D}_m} \left(y_n - \hat{g}_m\right)^2 \right\}$$

选择了第 j 个特征和 j 特征的取值 a_o 作为切分点

在根结点，按照 $x_j \le a_o$ 和 $x_j > a_o$ 分成左右两个分支

将样本集分成 $\mathbf{D}_1, \mathbf{D}_2$ ，并计算逼近误差

$$\varepsilon_m = \sqrt{\frac{1}{N_m} \sum_{(\boldsymbol{x}_n, y_n) \in \mathbf{D}_m} \left(y_n - \hat{g}_m\right)^2}, \; m = 1, 2$$

回归树：

对于预定门限 ε_0，若在子结点 T_1 满足 $\varepsilon_1 < \varepsilon_0$

则 T_1 设为叶结点，该结点的输出为 $\hat{g}_1$

否则 以 T_1 为子树的根结点，$\mathbf{D}_1$ 为样本集

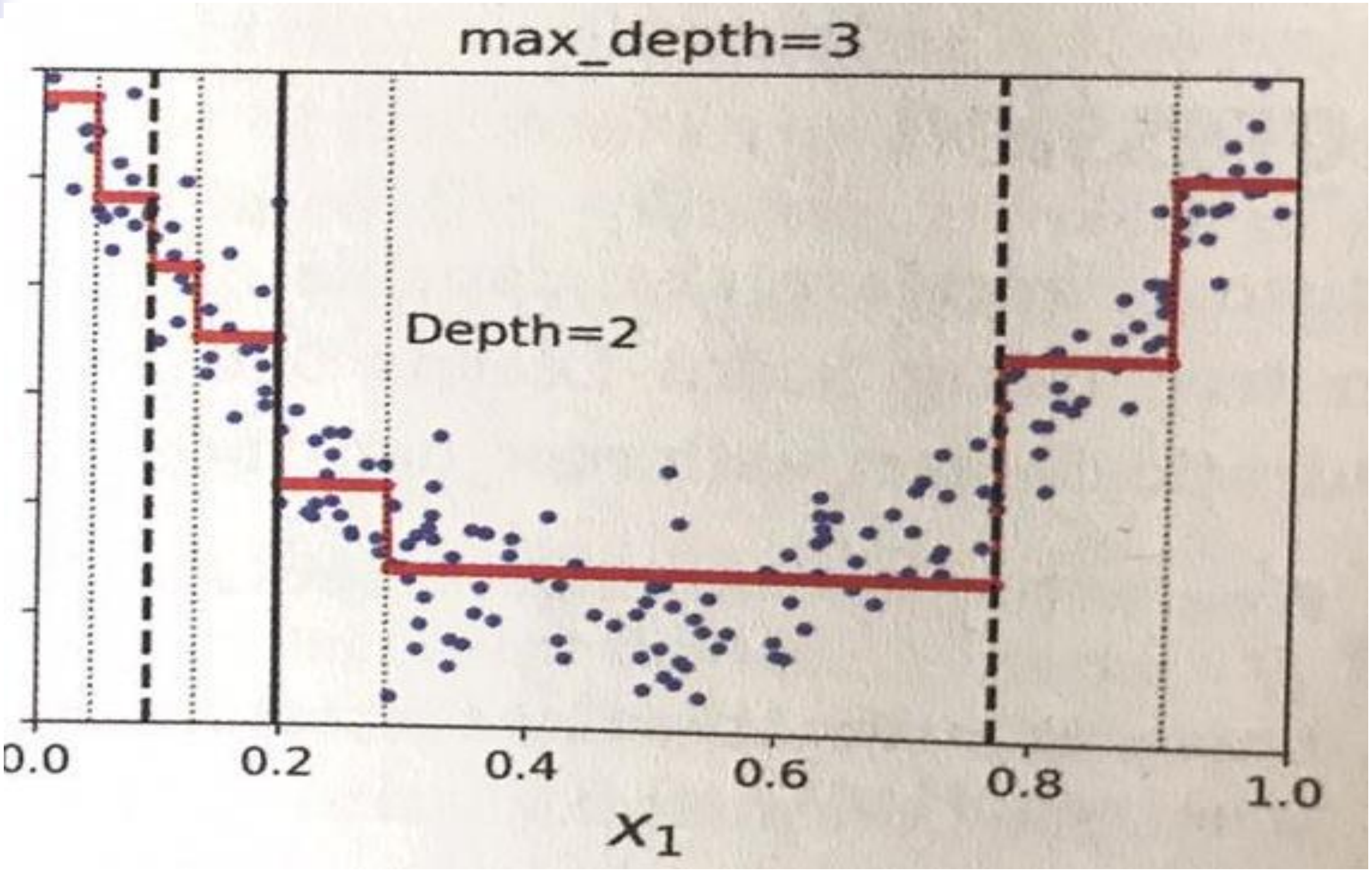
递归进行操作，对 T_2 也是同样过程

直到所有结点都终结在叶结点

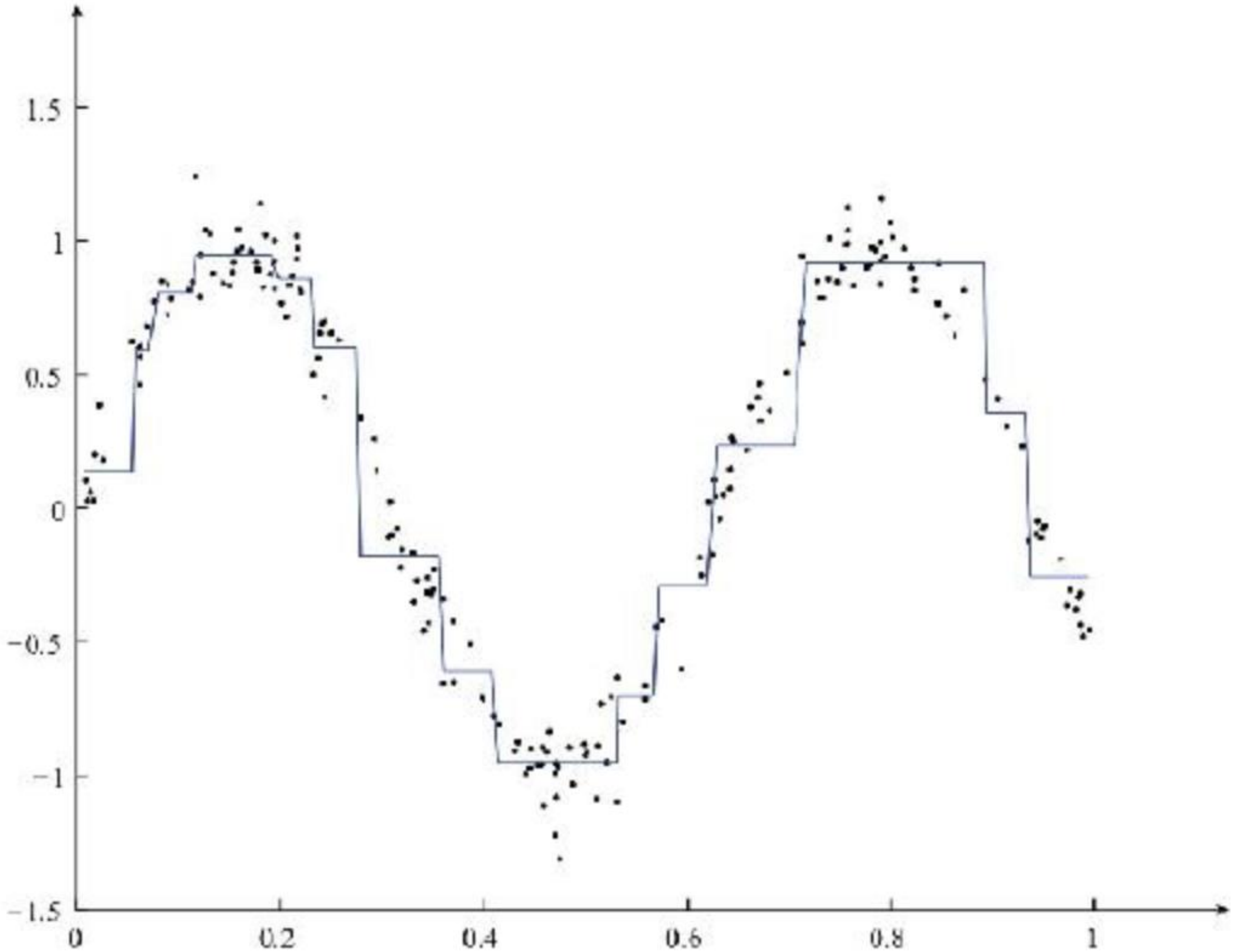
当回归树完成后，设共形成 M 个叶结点 M 个区域

回归模型为 $$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{M} \hat{g}_i I(\boldsymbol{x} \in \mathbf{R}_i)$$

回归树的一个数值实例



一个更深回归树的实例

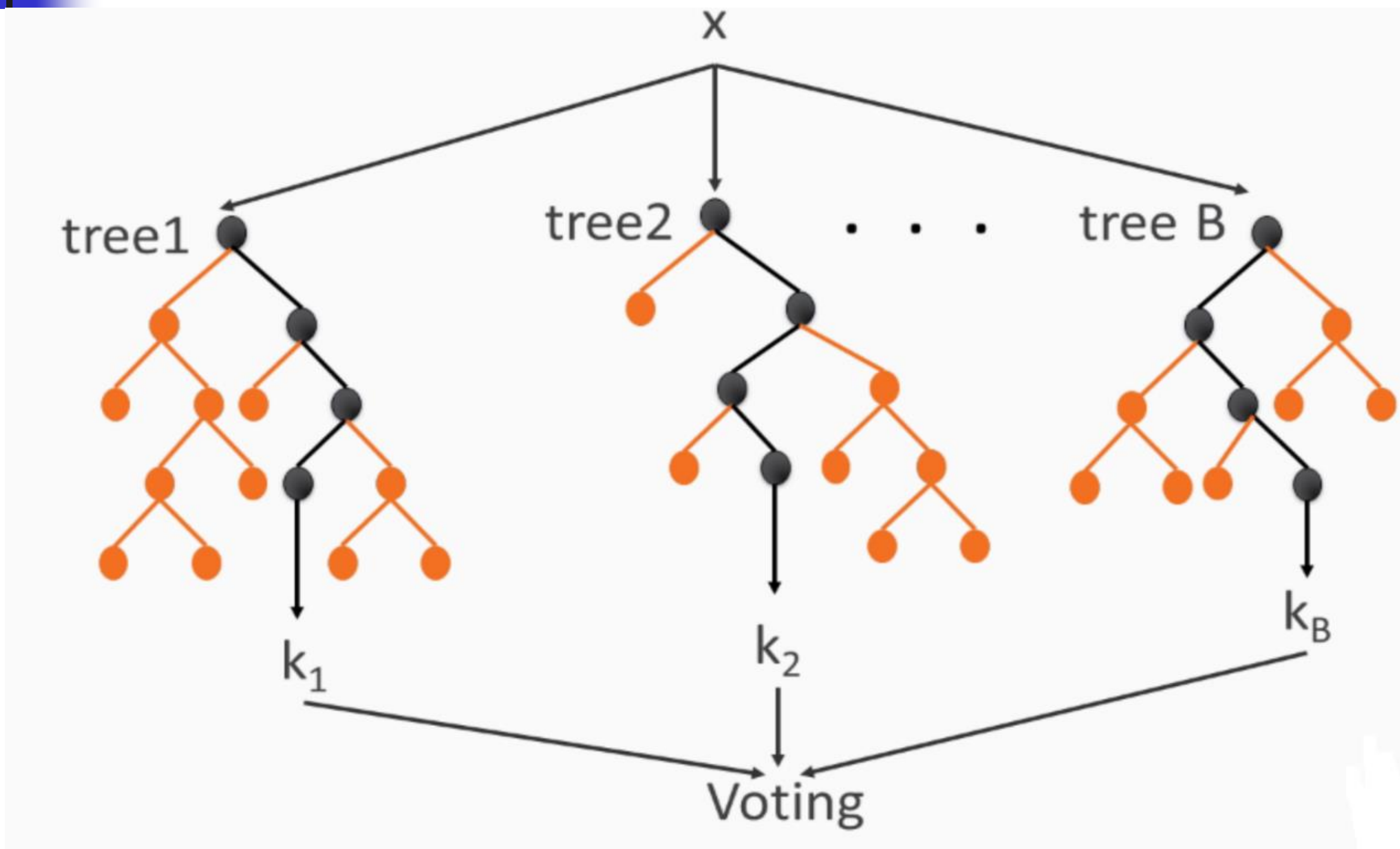


5. 决策树的一些扩展问题

- 连续数值变量
- 正则化和剪枝技术
  - 预剪枝技术
  - 后剪枝技术
- 缺少属性的训练样本问题
- 逻辑表达式和解释性

6. 集成学习-1： Bagging和随机森林（Random Forests）

在决策树基础上的集成学习方法:对样本进行随机采样，对采样样本生成决策树，多决策树组合。 Breiman 2001



6.1 自助采样和Bagging算法

- 自助采样（bootstrap）
  - 原始训练样本集为 $\boldsymbol{D}=\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$

自助采样是指，随机的从 $\boldsymbol{D}$ 中抽取一个样本放入 $\boldsymbol{D}^*$ 中，

同时将该样本放回 $\boldsymbol{D}$ 中，按照这个方式采样 N 次，

组成自助样本集 $\boldsymbol{D}^*$ 。

可获得 B 个自助样本集，记为 $\boldsymbol{D}^{*(b)}, b=1,2,\cdots,B$ 。

假设 N 充分大, $(\boldsymbol{x}_k, y_k)$ 被包含在样本集 $\boldsymbol{D}^{*(j)}$ 中的概率为

$$1-\left(1-\frac{1}{N}\right)^N \approx 1-\mathrm{e}^{-1} \approx 0.632$$

Bagging（bootstrap aggregation）算法

(1) 由训练样本集 $\boldsymbol{D}$，重采样得到 B 个自助样本集 $\boldsymbol{D}^{*(b)}, b=1,2,\cdots,B$

(2) 对于每一个 $\boldsymbol{D}^{*(b)}$，通过基学习算法训练一个基学习器 $\hat{f}^{*(b)}(\boldsymbol{x})$

(3) 则 Bagging 集成学习器为

$$\hat{f}_{bag}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*(b)}(\boldsymbol{x})$$

特点：

在每个基学习器 $\hat{f}^{*(b)}(\boldsymbol{x})$ 是一种“不稳定”学习器时

Bagging 可显著降低集成学习器的方差。

6.2 随机森林算法描述（Random Forests）

通过进一步增加随机性：随机选取特征，提高有效性

1. 对于 $b=1,2,\cdots,B$

   (a) 通过自举采样，从训练集中得到 N 个样本的集合 $\mathbf{X}_b$

   (b) 利用自举样本集生成一棵随机森林树 T_b，在树的每个节点通过递归重复如下步骤，直到达到最小节点规模 $n_{\min}$

   (I) 从 D 个特征变量中随机选取 m 个变量

   (II) 在 m 个变量中选择最好的变量和切分点

   (III) 分裂结点到两个子结点

2. 输出树的集合 $\{T_b\}_{b=1}^{B}$

随机森林做预测

对于新的输入 $\boldsymbol{x}$ 做预测，分别：

回归： $\hat{y}_{rf}^{B}(\boldsymbol{x})=\frac{1}{B}\sum_{b=1}^{B}T_b(\boldsymbol{x})$

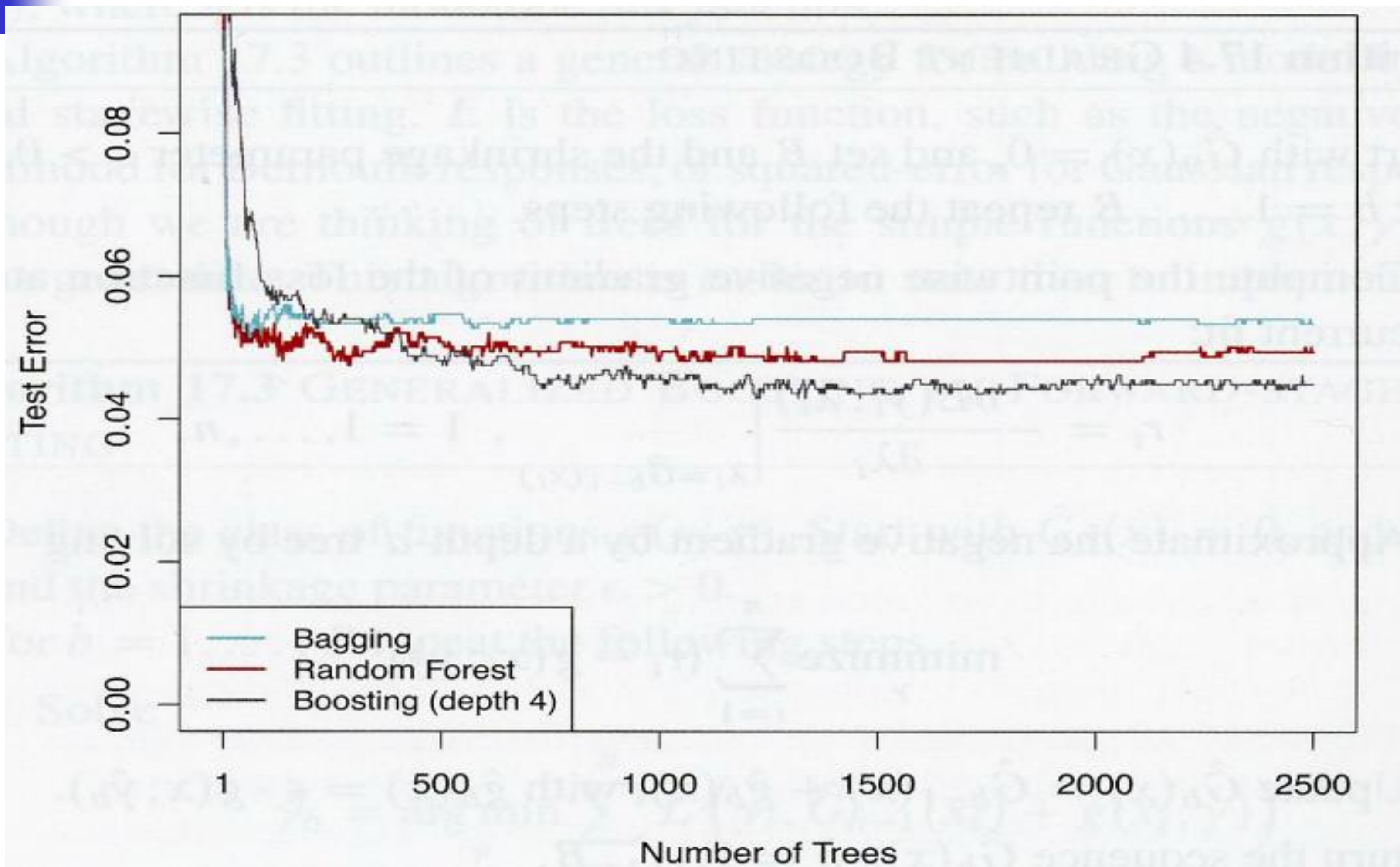
（$T_b(\boldsymbol{x})$ 表示树 $T_b(\boldsymbol{x})$ 的回归输出）

分类：投票原则， $\hat{C}_{rf}^{B}(\boldsymbol{x})=vote\left\{\hat{C}_{rf}^{b}(\boldsymbol{x})\right\}_{b=1}^{B}$

（$\hat{C}_{rf}^{b}(\boldsymbol{x})$ 是树 T_b 的类型输出）

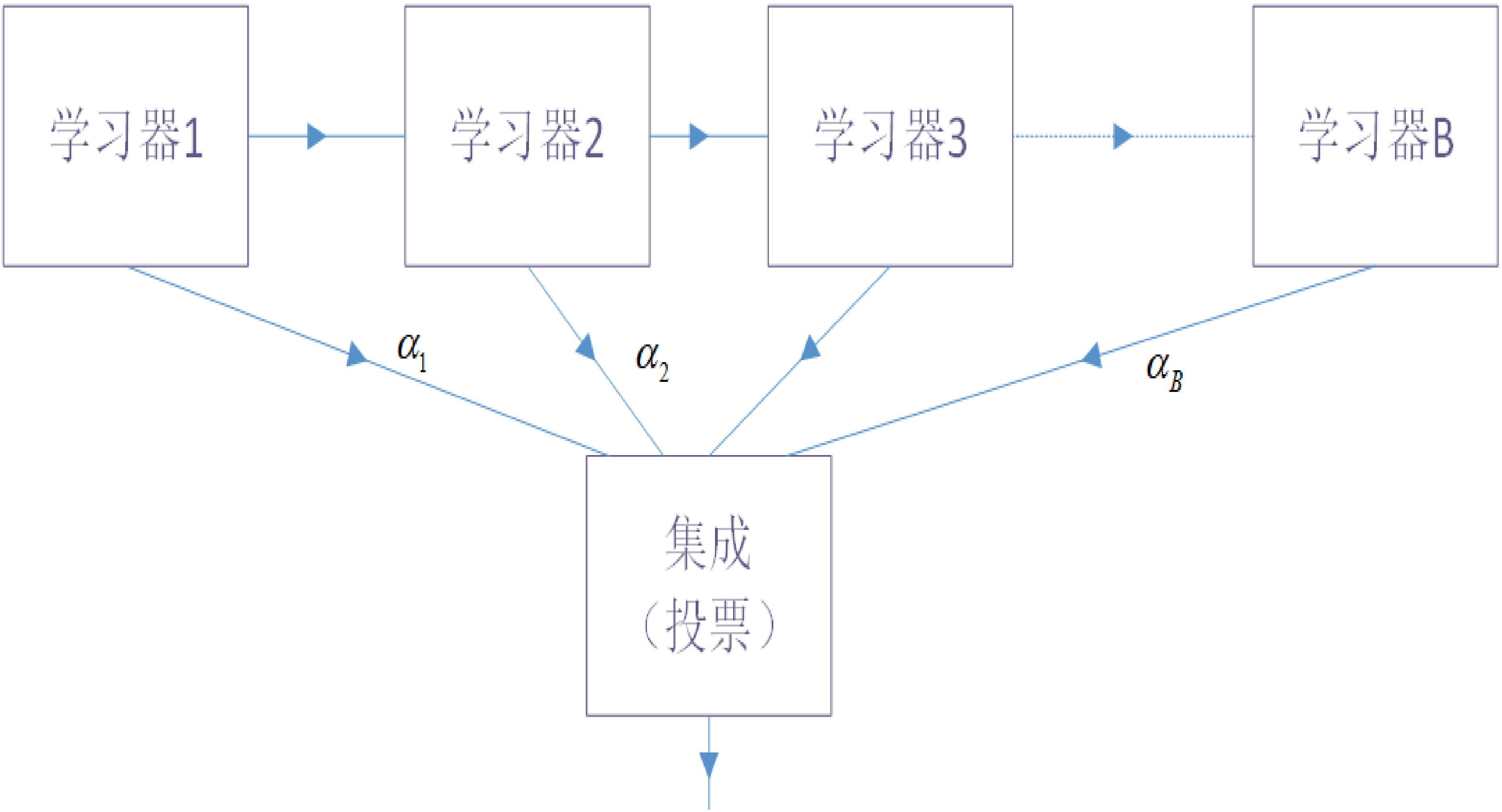
性能比较

对spam数据集，Bagging、随机森林和提升树性能比较



7. 集成学习-2：提升和AdaBoost算法

提升（boosting）类算法，组合若干个弱学习器构成一个强学习器。（串行结构）



AdaBoost算法

AdaBoost由Freund和Schapiro于1995年提出

基本的AdaBoost算法用于2分类问题，以 $\{-1,+1\}$ 表示两类

AdaBoost算法的基本思想：
（1）对基学习器进行多轮调用；
（2）在每一轮调用时，都对样本集中每个样本在损失函数中的权重进行调整
（3）初始时所有样本具有相等的权重，但在经过每一轮，被正确分类的样本给予较小权重，没有被正确分类的样本权重增加
（4）难以正确分类的样本会持续获得高权重，使得后续基学习器重点关注和解决较难分类的样本。

AdaBoost算法描述

给定训练样本集 $\boldsymbol{D}=\left\{\left(\boldsymbol{x}_n, y_n\right)\right\}_{n=1}^{N}$，$\boldsymbol{x}_n \in \boldsymbol{X}$，$y_n \in \{-1,+1\}$

初始化分布：$D_1(n)=\dfrac{1}{N}$，$n=1,2,\cdots,N$

对于 $t=1,2,\cdots,B$

根据分布 D_t 训练弱分类器，得到弱分类器 $h_t:\boldsymbol{X}\rightarrow\{-1,+1\}$

目标：选择 h_t 使得加权后的误差 ε_t 最小，ε_t 为

$$\varepsilon_t \doteq \mathrm{P}_{n\sim D_t}\left(h_t(\boldsymbol{x}_n)\neq y_n\right)=\sum_{n=1}^{N} D_t(n) I\left(h_t(\boldsymbol{x}_n)\neq y_n\right)$$

AdaBoost算法描述（续）

取 $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

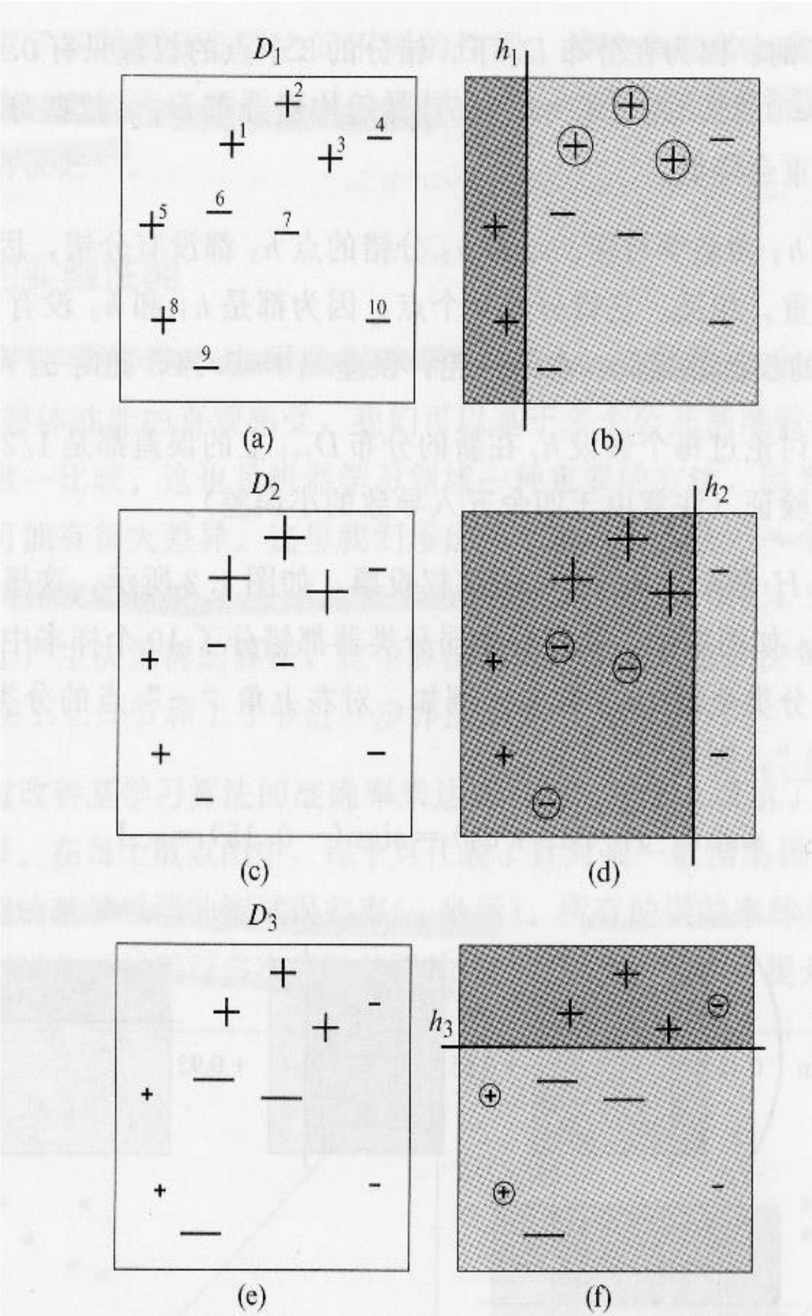
进行如下更新，对 $n = 1, 2, \cdots, N$

$$D_{t+1}(n) = \frac{D_t(n)}{Z_t}\exp\left(-\alpha_t y_n h_t(\boldsymbol{x}_n)\right)$$

这里，Z_t 是归一化因子，保证 D_{t+1} 是一个分布。

输出最终的学习器

$$H(\boldsymbol{x}) = \operatorname{sgn}\left[\sum_{t=1}^{B}\alpha_t h_t(\boldsymbol{x})\right]$$

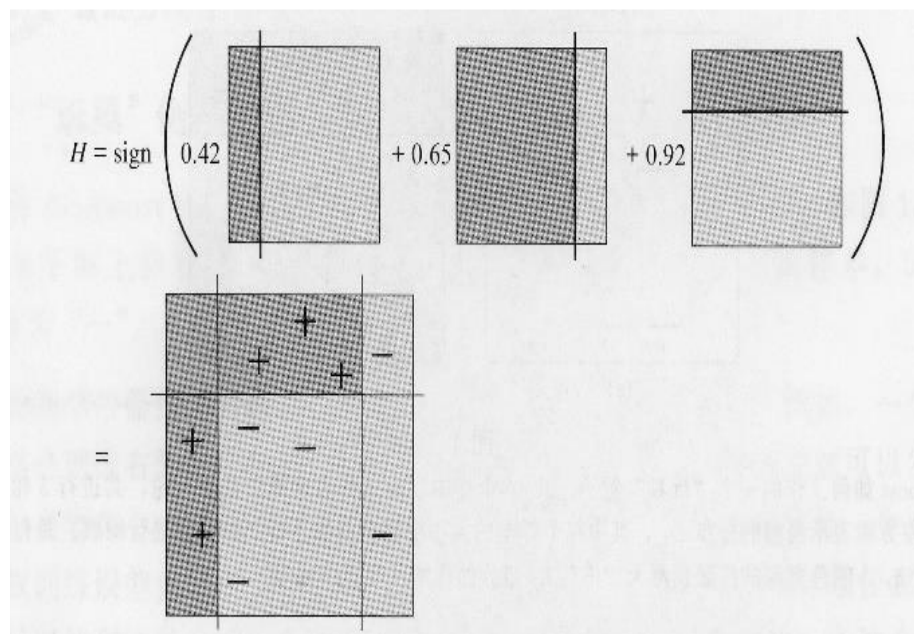


AdaBoost算法的示例：

三轮提升，得到对所有样本正确分类的集成分类器。

图中分别用“+”和“-”表示样本的类型

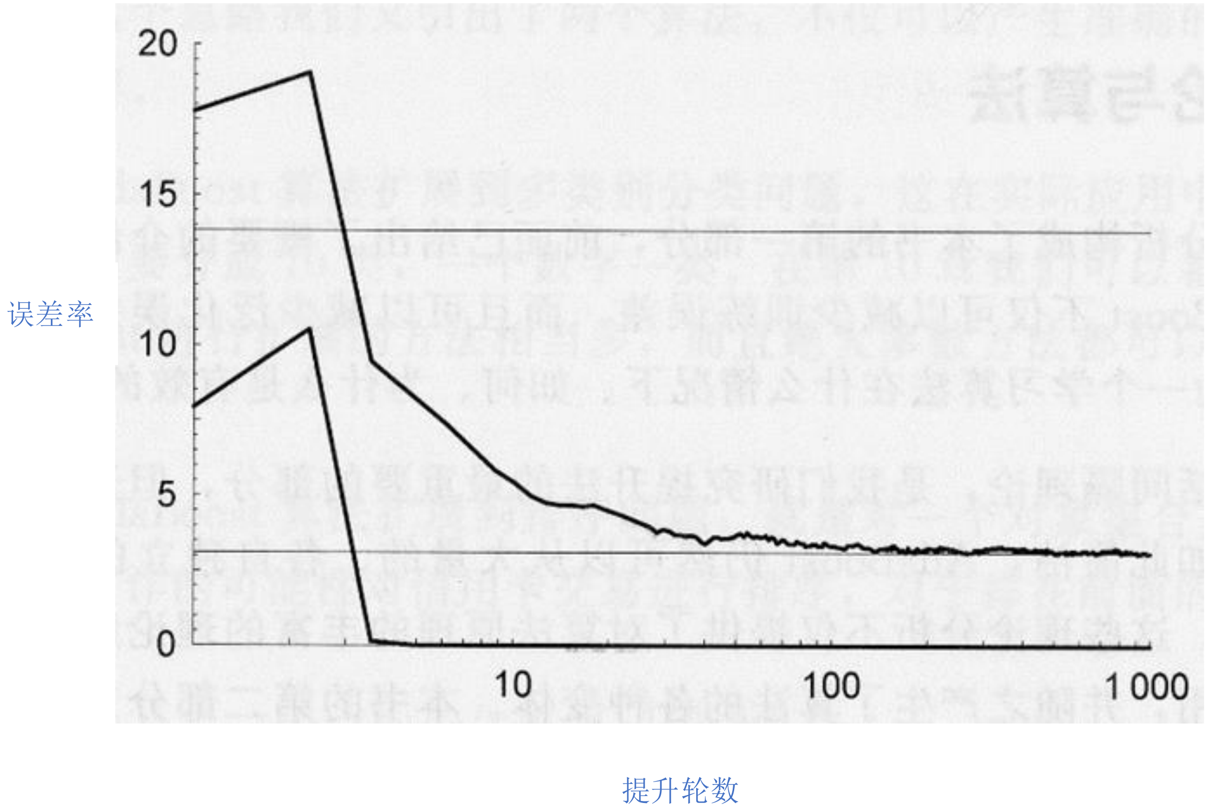
$$H(\boldsymbol{x}) = \operatorname{sgn}\left[0.42h_1(\boldsymbol{x}) + 0.65h_2(\boldsymbol{x}) + 0.92h_3(\boldsymbol{x})\right]$$



AdaBoost的泛化性能评价

对于手写英文字母（OCR）数据集（16000训练样本，4000测试样本），使用C4.5决策树做基分类器

利用AdaBoost集成方法，随提升数增加时训练误差和测试误差的变化曲线



8. 加法模型和提升树算法

加法模型表示为

$$F(\boldsymbol{x})=\sum_{t=1}^{B}\beta_t b(\boldsymbol{x};\theta_t)$$

定义目标函数 $L\big(y,F(\boldsymbol{x})\big)$

加法模型优化问题为

$$\min_{\{\beta_t,\theta_t\}_{t=1}^{B}}\left\{\sum_{n=1}^{N}L\left(y_n,\sum_{t=1}^{B}\beta_t b(\boldsymbol{x}_n;\theta_t)\right)\right\}$$

加法模型和提升树算法（续）

模型分步更新

$$F_t(\boldsymbol{x})=F_{t-1}(\boldsymbol{x})+\beta_t b(\boldsymbol{x};\theta_t)$$

前向一步优化问题简化为

$$(\beta_t,\theta_t)=\arg\min_{\beta,\theta}\left\{\sum_{n=1}^{N}L\left(y_n,F_{t-1}(\boldsymbol{x}_n)+\beta b(\boldsymbol{x}_n;\theta)\right)\right\}$$

基学习器 $b(\boldsymbol{x};\theta_t)$ 采用决策树，则构成提升树

一棵决策树的模型表示

$$T(\boldsymbol{x};\Theta)=\sum_{j=1}^{J}c_j I\left(\boldsymbol{x}\in R_j\right)$$

决策树的参数：$\Theta=\{R_j,c_j\}_{j=1}^{J}$ ，J是树深度

加法模型和提升树算法（续）

提升树模型表示为

$$F_B(\boldsymbol{x}) = \sum_{t=1}^{B} T(\boldsymbol{x};\Theta_t)$$

第t轮决策树参数的确定为如下优化问题

$$\hat{\Theta}_t = \arg\min_{\Theta_t} \left\{ \sum_{n=1}^{N} L\left(y_n, F_{t-1}(\boldsymbol{x}_n) + T(\boldsymbol{x}_n;\Theta_t)\right) \right\}$$

第t轮参数：$\Theta_t = \{R_{tj}, c_{tj}\}_{j=1}^{J_t}$

实现示例-1：指数函数和AdaBoost

取损失函数：$L\left(y_n, F(x_n)\right) = \mathrm{e}^{-y_n F(x_n)}$ 提升树为Adaboost

加法模型和提升树算法（续）

实现示例-2：平方误差函数-回归提升树

对于回归问题，取

$$L\big(y,F(\boldsymbol{x})\big)=\frac{1}{2}\big[y-F(\boldsymbol{x})\big]^2$$

参数优化问题为

$$\begin{aligned}\hat{\Theta}_t &= \arg\min_{\Theta_t}\left\{\sum_{n=1}^{N} L\big(y_n, F_{t-1}(\boldsymbol{x}_n)+T(\boldsymbol{x}_n;\Theta_t)\big)\right\}\\ &= \arg\min_{\Theta_t}\left\{\sum_{n=1}^{N}\frac{1}{2}\big[y_n-F_{t-1}(\boldsymbol{x}_n)-T(\boldsymbol{x}_n;\Theta_t)\big]^2\right\}\\ &= \arg\min_{\Theta_t}\left\{\sum_{n=1}^{N}\frac{1}{2}\big[r_{tn}-T(\boldsymbol{x}_n;\Theta_t)\big]^2\right\}\end{aligned}$$

定义残差： $r_{tn}=y_n-F_{t-1}(\boldsymbol{x}_n)$

第t轮决策树，目标函数以残差替代标注值训练一棵决策树

回归提升树算法

引入了收缩参数！

输入：训练数据集 $\boldsymbol{D}=\{(\boldsymbol{x}_n,y_n)\}_{n=1}^{N}$；给出收缩因子 $0<\varepsilon\leq 1$

初始化：$F_0(\boldsymbol{x})=0$，$r_n=y_n,\ n=1,2,\cdots,N$

对于 $t=1,2,\cdots,B$

以 r_n 为标注，学习一棵回归树 $T(\boldsymbol{x};\Theta_t)$（树结点数 J_t）

得到收缩决策树：$\hat{T}(\boldsymbol{x};\Theta_t)=\varepsilon T(\boldsymbol{x};\Theta_t)$

集成决策树：$F_t(\boldsymbol{x})=F_{t-1}(\boldsymbol{x})+\hat{T}(\boldsymbol{x};\Theta_t)$

更新残差：$r_n\leftarrow r_n-\hat{T}(\boldsymbol{x}_n;\Theta_t),\ n=1,2,\cdots,N$

输出提升树：$F_B(\boldsymbol{x})$

9. 梯度提升树算法
（gradient boosting decision tree, GBDT）

在一般情况下，定义新的等价残差为负梯度

$$r_{tn} = -g_{tn} = -\left[\frac{\partial L\left(y_n, F(\boldsymbol{x}_n)\right)}{\partial F(\boldsymbol{x}_n)}\right]_{F(\boldsymbol{x}_n)=F_{t-1}(\boldsymbol{x}_n)}$$

引入收缩参数

$$0 < \varepsilon \leq 1$$

对于一系列损失函数，引入梯度提升决策树算法并应用于回归和分类

（回归的）梯度提升决策树

输入：训练数据集 $\boldsymbol{D}=\left\{\left(\boldsymbol{x}_n, y_n\right)\right\}_{n=1}^{N}$，给出目标函数 $L(\cdot,\cdot)$

给出收缩因子 $0<\varepsilon\le 1$

初始化：$F_0(\boldsymbol{x})=\arg\min\limits_{c}\sum_{n=1}^{N}L\left(y_n, c\right)$

对于 $t=1,2,\cdots,B$

(a) 对于 $n=1,2,\cdots,N$，计算

$$r_{tn}=-\left[\frac{\partial L\left(y_n, F(\boldsymbol{x}_n)\right)}{\partial F(\boldsymbol{x}_n)}\right]_{F(\boldsymbol{x}_n)=F_{t-1}(\boldsymbol{x}_n)}$$

(b) 训练一个回归树去拟合标注 r_{tn}，得到叶结点和对应的区域

$R_{tj}, j = 1, 2, \cdots, J_t$

(c) 对于 $j = 1, 2, \cdots, J_t$，计算

$$c_{tj} = \arg\min_{c} \left\{ \sum_{\boldsymbol{x}_n \in R_{tj}} L\left(y_n, F_{t-1}(\boldsymbol{x}_n) + c\right) \right\}$$

(d) 更新集成决策树：$F_t(\boldsymbol{x}) = F_{t-1}(\boldsymbol{x}) + \varepsilon \sum_{j=1}^{J_t} c_{tj} I\left(\boldsymbol{x} \in R_{tj}\right)$

输出提升树：$F_B(\boldsymbol{x})$

决策树和集成学习小结

- 决策树简单、可解释性强，但性能不高
- 集成学习可有效提升单学习器的性能
- 以决策树为基本学习器构成随机森林，是一种高效、并行性强的集成算法
- AdaBoost是一种有效的串行分类提升算法
- 提升树尤其梯度提升树（GBDT）构成一类广泛的集成算法
- GBDT的增强版，XGBoost或LightGBM等可在性能或有效性方面进一步提升（可参考其网络社区）。