决策树、随机森林、adaboost

七月算法 龙老师

2016年5月28日

主要内容

- 复习信息熵
  - 熵、互信息
- 决策树学习算法
  - 信息增益
  - ID3、C4.5、CART
- Bagging与随机森林
- 提升
  - Adaboost/GDBT

熵与互信息

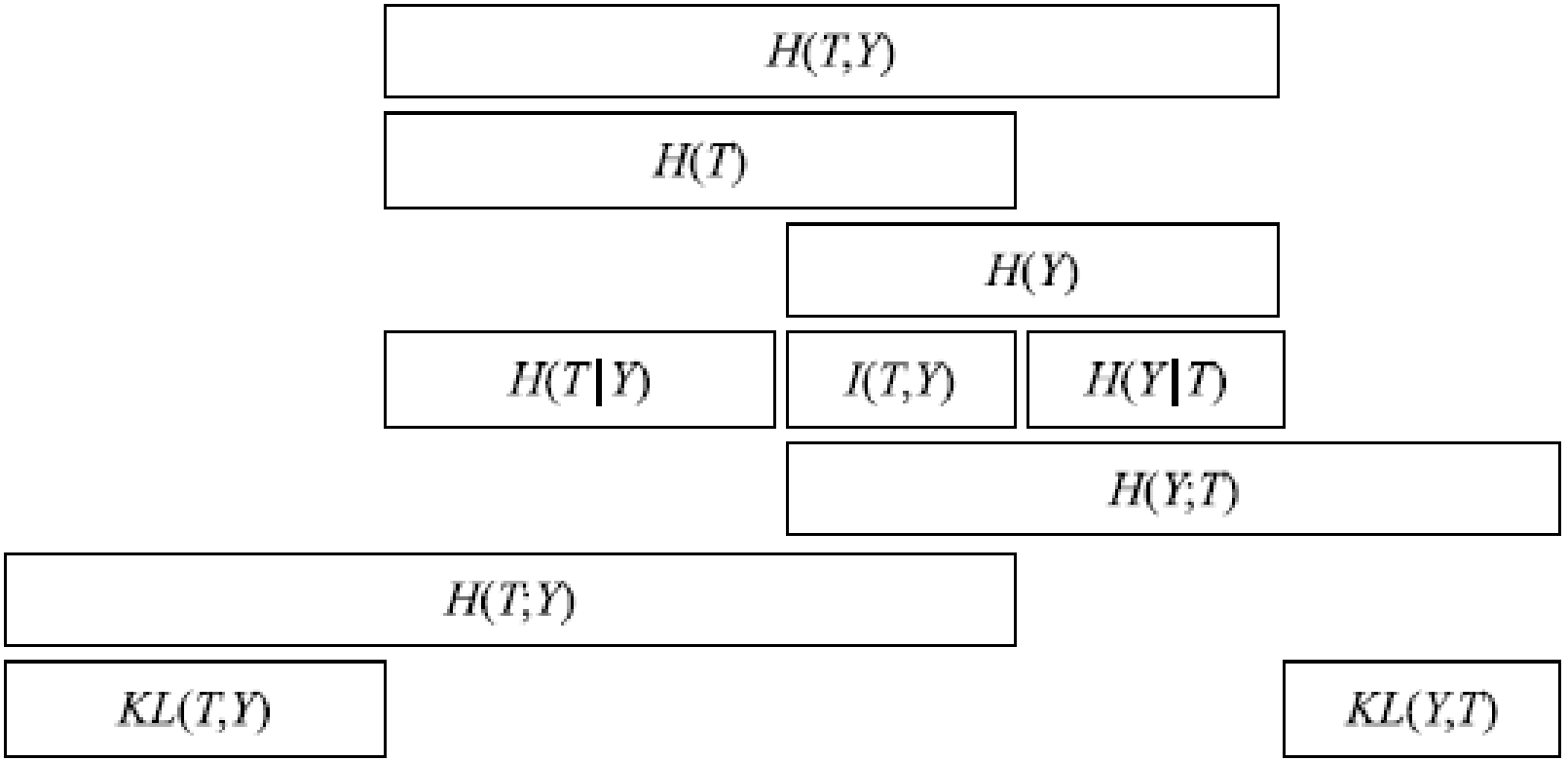
- 熵是对平均不确定性的度量

$$H(X) = -\sum_{x \in X} P(x) \cdot \log P(x)$$

- 平均互信息：得知特征Y的信息而使得对标签X的信息的不确定性减少的程度。

$$I(X;Y) = \sum_{x \in X, y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

熵与互信息



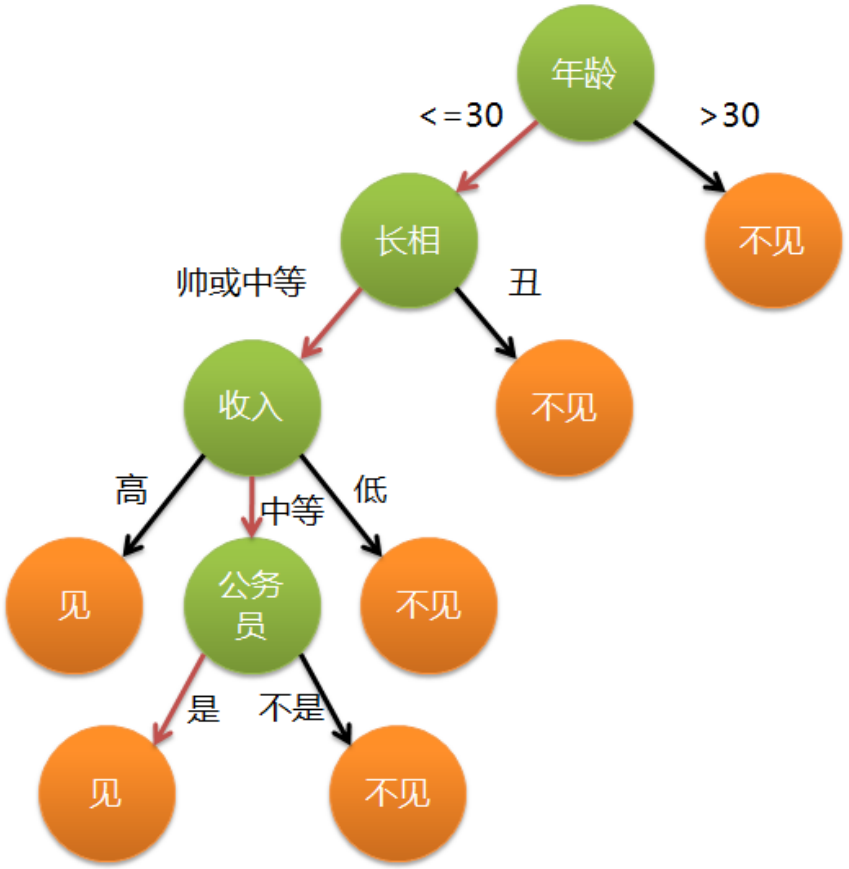
各个熵之间的关系

Table 1. Some information formulas and their properties as learning measures.

Name	Formula	(Dis)similarity	(A)symmetry
Joint Information	$H(T,Y) = -\sum_t \sum_y p(t,y) \log_2 p(t,y)$	Inapplicable	Symmetry
Mutual Information	$I(T,Y) = \sum_t \sum_y p(t,y) \log_2 \frac{p(t,y)}{p(t)p(y)}$	Similarity	Symmetry
Conditional Entropy	$H(Y\|T) = -\sum_t \sum_y p(t,y) \log_2 p(y\|t)$	Dissimilarity	Asymmetry
Cross Entropy	$H(T;Y) = -\sum_z p_t(z) \log_2 p_y(z)$	Dissimilarity	Asymmetry
KL Divergence	$KL(T,Y) = \sum_z p_t(z) \log_2 \frac{p_t(z)}{p_y(z)}$	Dissimilarity	Asymmetry

决策树

- 根节点
- 父节点
- 子节点
- 叶子节点
- 分叉
- 属性
- 标签



决策树(Decision Tree)

- 决策树学习采用的是自顶向下的递归方法，
- 其基本思想是以信息熵为度量构造一棵熵值下降最快的树，到叶子节点处的熵值为零，
- 此时每个叶节点中的实例都属于同一类。
- 有监督学习

决策树学习的生成算法

- 建立决策树的关键，即在当前状态下选择哪个属性作为分类依据。根据不同的目标函数，建立决策树主要有一下三种算法。
  - ID3
  - C4.5
  - CART

信息增益

- 特征A对训练数据集D的信息增益g(D,A)，定义为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差，即：
  - $g(D,A)=H(D) - H(D|A)$
  - 显然，这即为训练数据集D和特征A的互信息。
- 遍历所有特征，选择信息增益最大的特征作为当前的分裂特征

其他目标

- 信息增益率： gr(D,A) = g(D,A) / H(A)
  - C4.5
- Gini系数：
  - CART

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

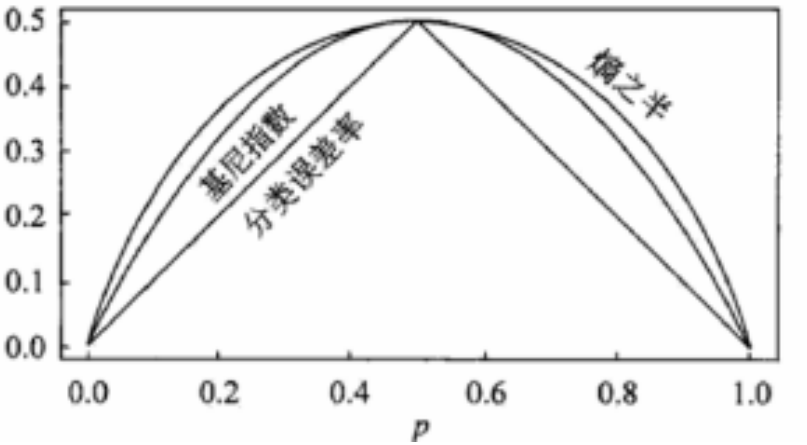
$$=1-\sum_{k=1}^{K}\left(\frac{|C_k|}{|D|}\right)^2$$

关于Gini系数的讨论

- 考察Gini系数的图像、熵、分类误差率三者之间的关系
  - 将f(x)=-lnx在x=1处一阶展开，忽略高阶无穷小，得到f(x)≈1-x

$$H(X)=-\sum_{k=1}^{K} p_k \ln p_k$$

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



三种决策树学习算法

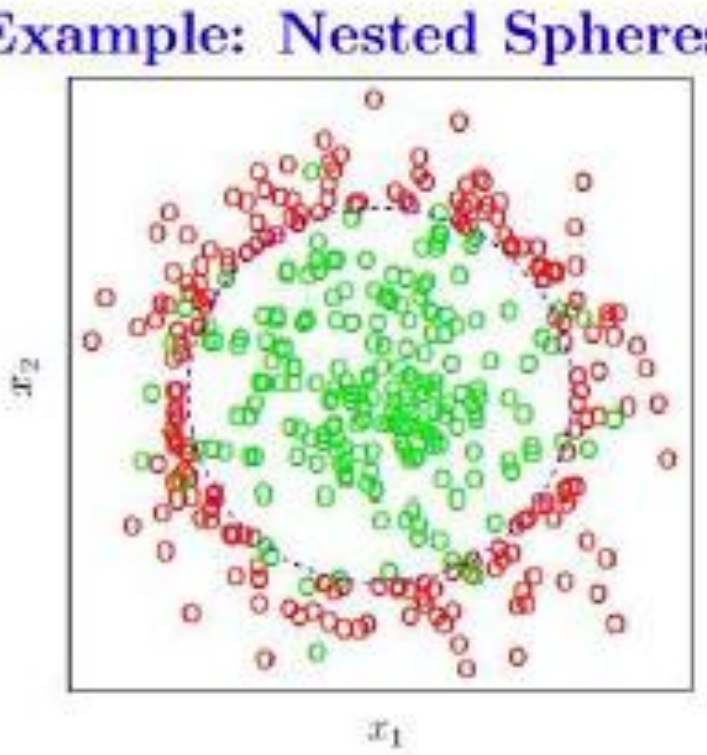
- ID3：
  - 取值多的属性，更容易使数据更纯，其信息增益更大。
  - 训练得到的是一棵庞大且深度浅的树：不合理。
- C4.5
- CART
  - 一个属性的信息增益(率)/gini指数越大，表明属性对样本的熵减少的能力更强，这个属性使得数据由不确定性变成确定性的能力越强

决策树的评价

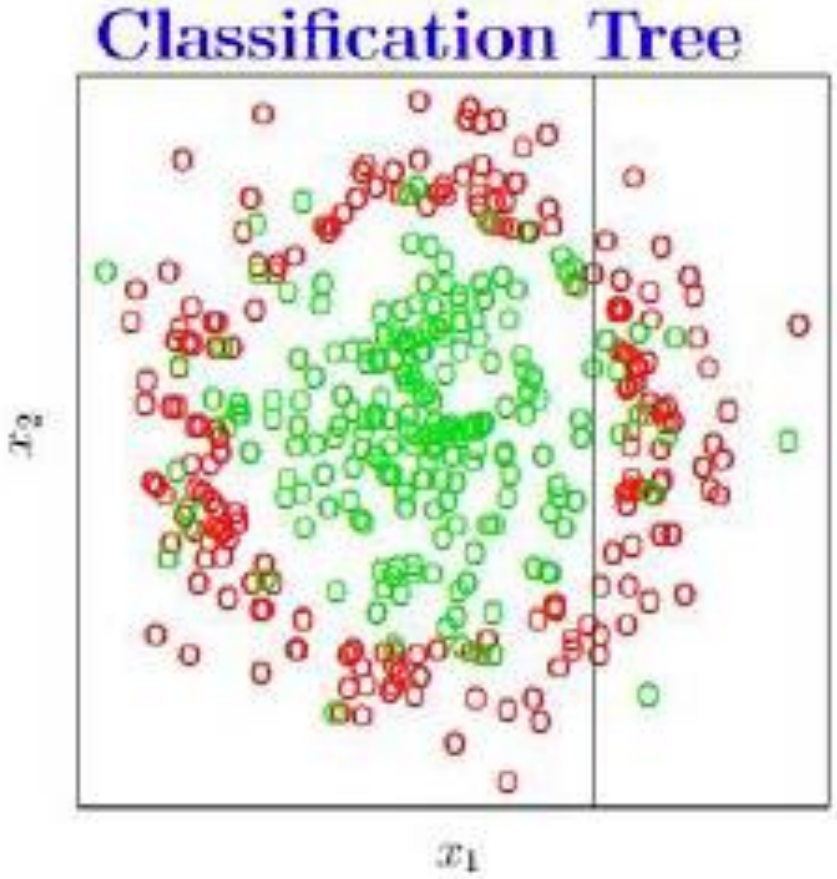
- 假定样本的总类别为K个。
- 对于决策树的某叶结点，假定该叶结点含有样本数目为n，其中第k类的样本点数目为n_k，k=1,2,…,K。
  - 若某类样本n_j=n而$n_1,\ldots,n_{j-1},n_{j+1},\ldots,n_K$=0，称该结点为纯结点；
  - 若各类样本数目$n_1=n_2=\ldots=n_k=n/K$，称该样本为均结点。
- 纯结点的熵H_p=0，最小
- 均结点的熵H_u=lnK，最大
- 对所有叶结点的熵求和，该值越小说明对样本的分类越精确。
  - 各叶结点包含的样本数目不同，可使用样本数加权求熵和
- 评价函数：$$C(T)=\sum_{t\in leaf} N_t \cdot H(t)$$
  - 由于该评价函数越小越好，所以，可以称之为“损失函数”。

决策树的例子

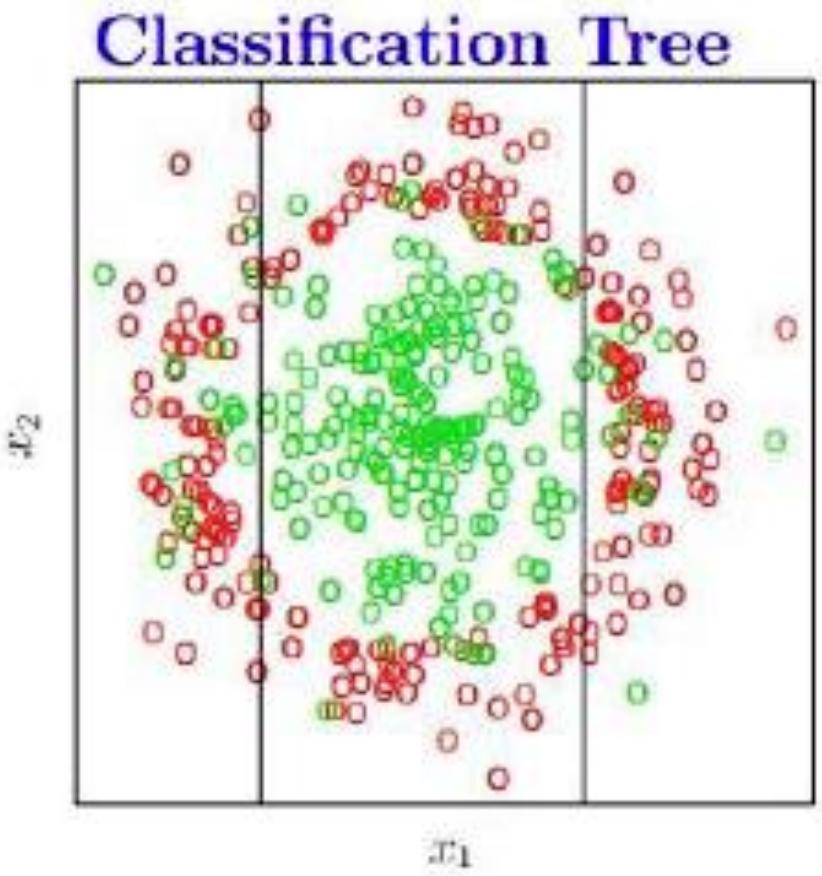
- 对于下面的数据，希望分割成红色和绿色两个类



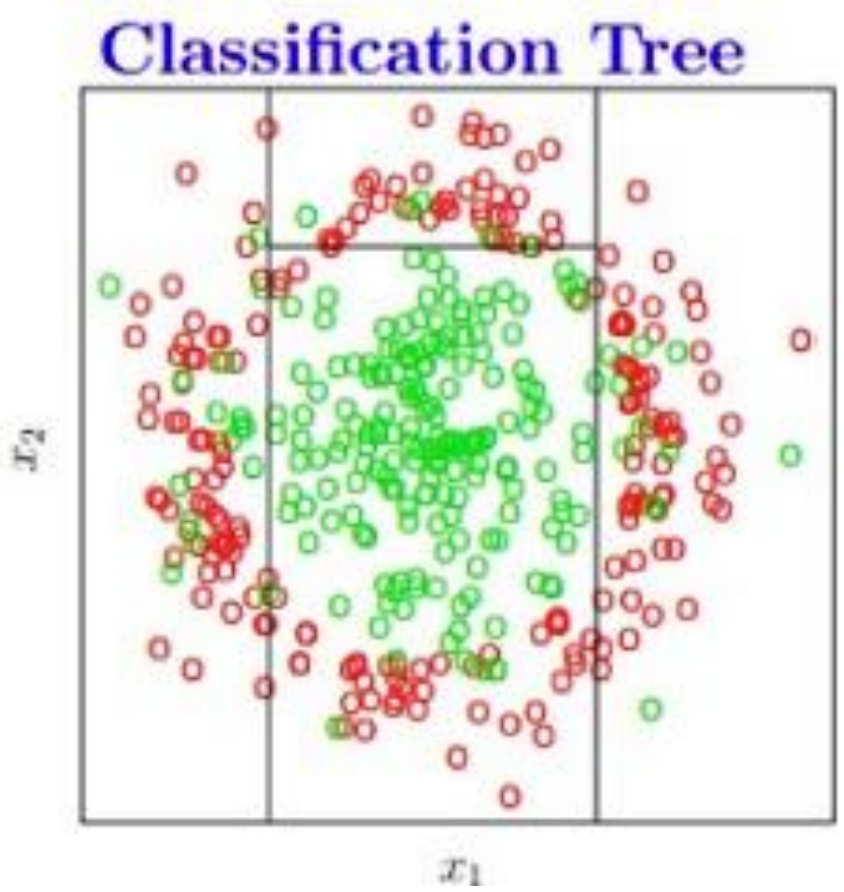
决策树的生成过程



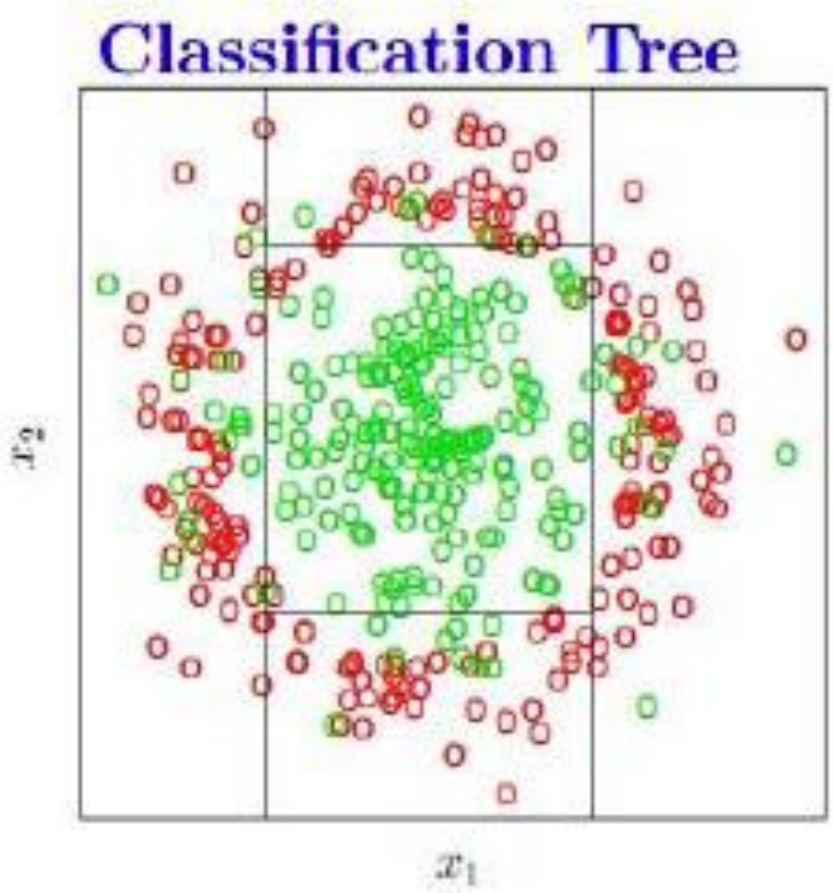
决策树的生成过程



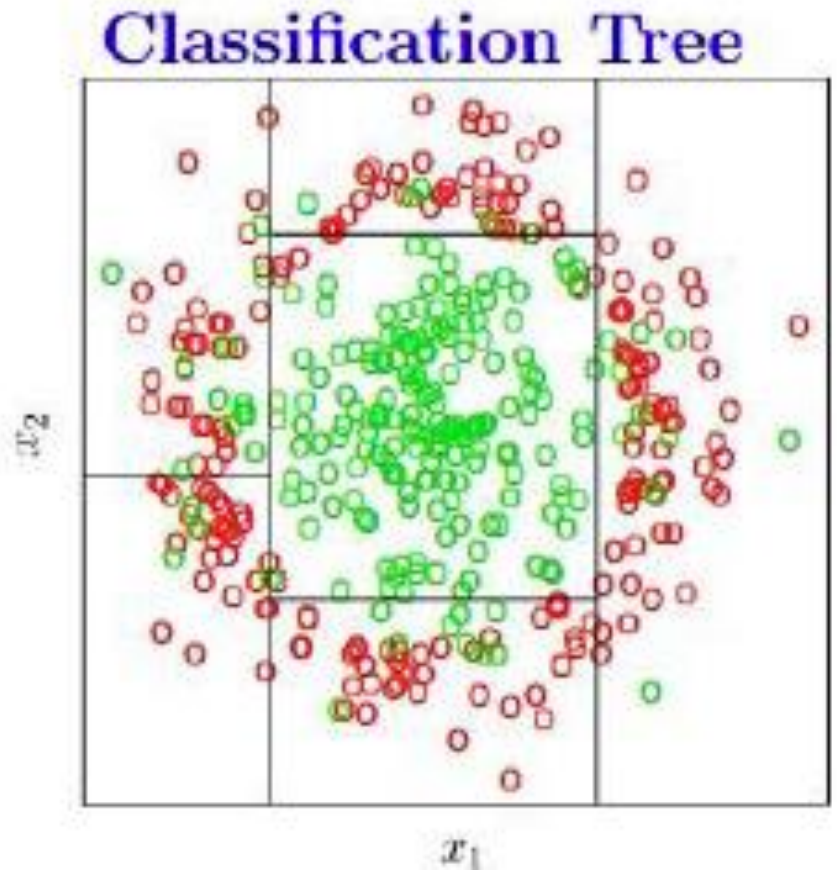
决策树的生成过程



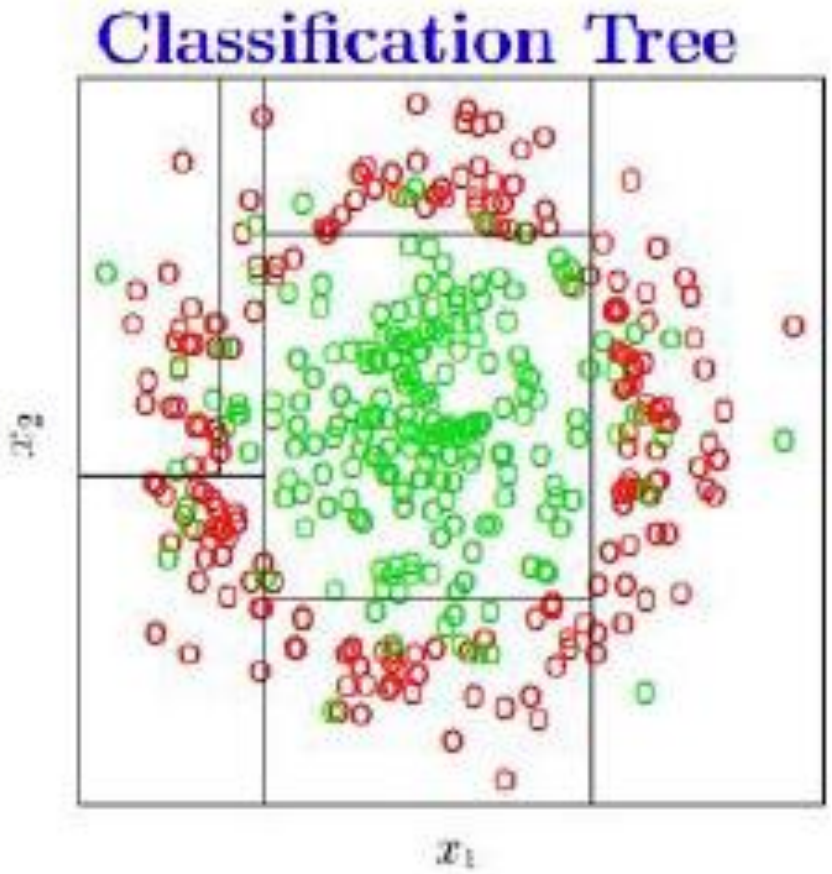
决策树的生成过程



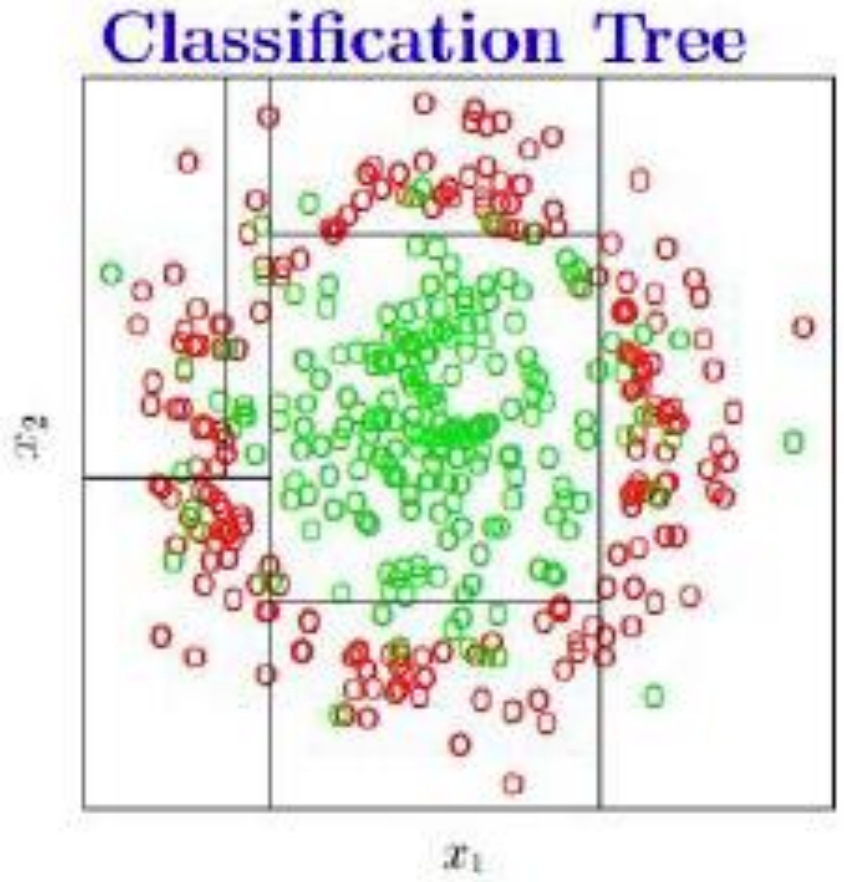
决策树的生成过程



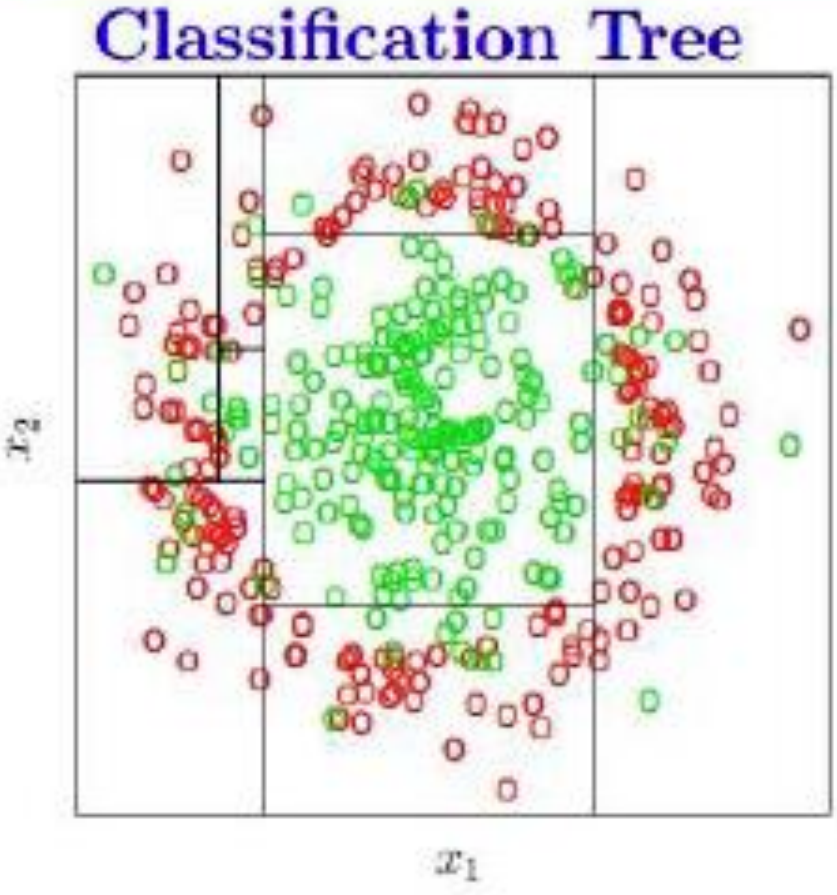
决策树的生成过程



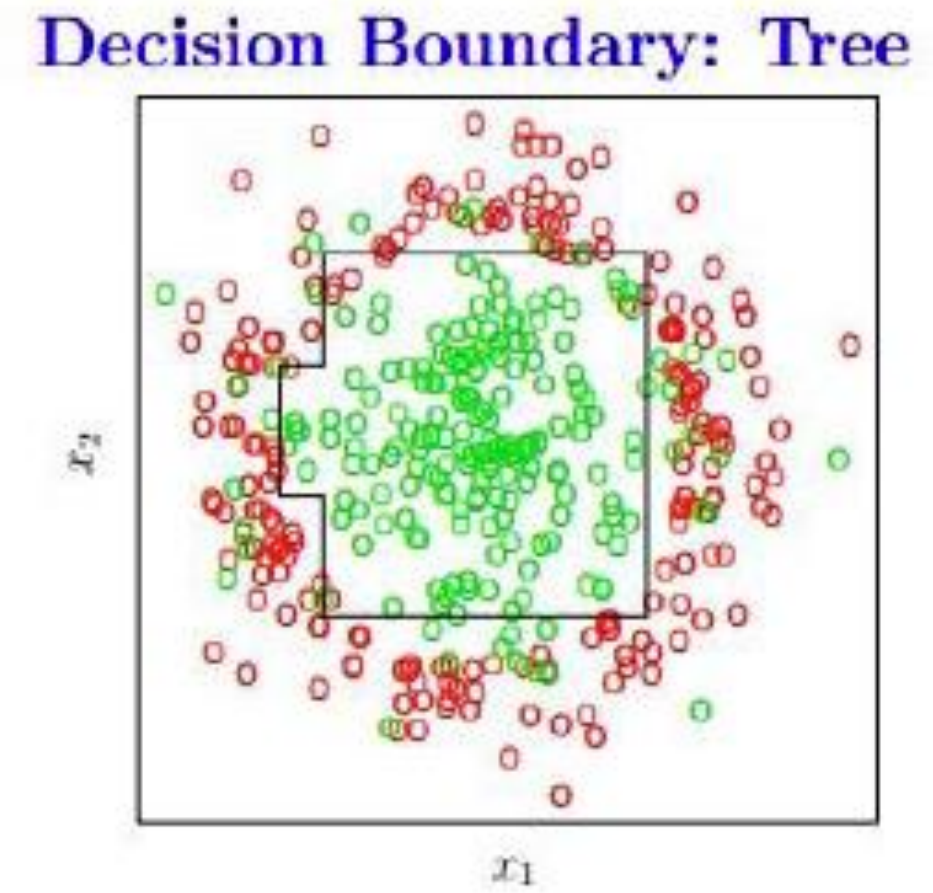
决策树的生成过程



决策树的生成过程



决策树的生成过程



决策树的过拟合

- 决策树对训练属于有很好的分类能力，但对未知的测试数据未必有好的分类能力，泛化能力弱，即可能发生过拟合现象。
  - 剪枝
  - 随机森林

剪枝

- 三种决策树的剪枝过程算法相同，区别仅是对于当前树的评价标准不同。
  - 信息增益、信息增益率、基尼系数
- 剪枝总体思路：
  - 由完全树T0开始，剪枝部分结点得到T1，再次剪枝部分结点得到T2...直到仅剩树根的树Tk；
  - 在验证数据集上对这k个树分别评价，选择损失函数最小的树Tα

剪枝系数的确定

- 根据原损失函数：$C(T)=\sum_{t\in leaf} N_t \cdot H(t)$
- 叶结点越多，决策树越复杂，损失越大，修正：
  - $C_\alpha(T)=C(T)+\alpha\cdot|T_{leaf}|$
  - 当α=0时，未剪枝的决策树损失最小；
  - 当α=+∞时，单根结点的决策树损失最小。
- 假定当前对以r为根的子树剪枝：
  - 剪枝后，只保留r本身而删掉所有的叶子
- 考察以r为根的子树：
  - 令剪枝后的损失函数=剪枝前的损失函数，求得a

剪枝算法

- 对于给定的决策树T0：
  - 计算所有内部节点的剪枝系数；
  - 查找最小剪枝系数的结点，剪枝得决策树Tk；
  - 重复以上步骤，直到决策树Tk只有1个结点；
  - 得到决策树序列T0T1T2...TK；
  - 使用验证样本集选择最优子树。
- 使用验证集做最优子树的标准，可以使用评价函数

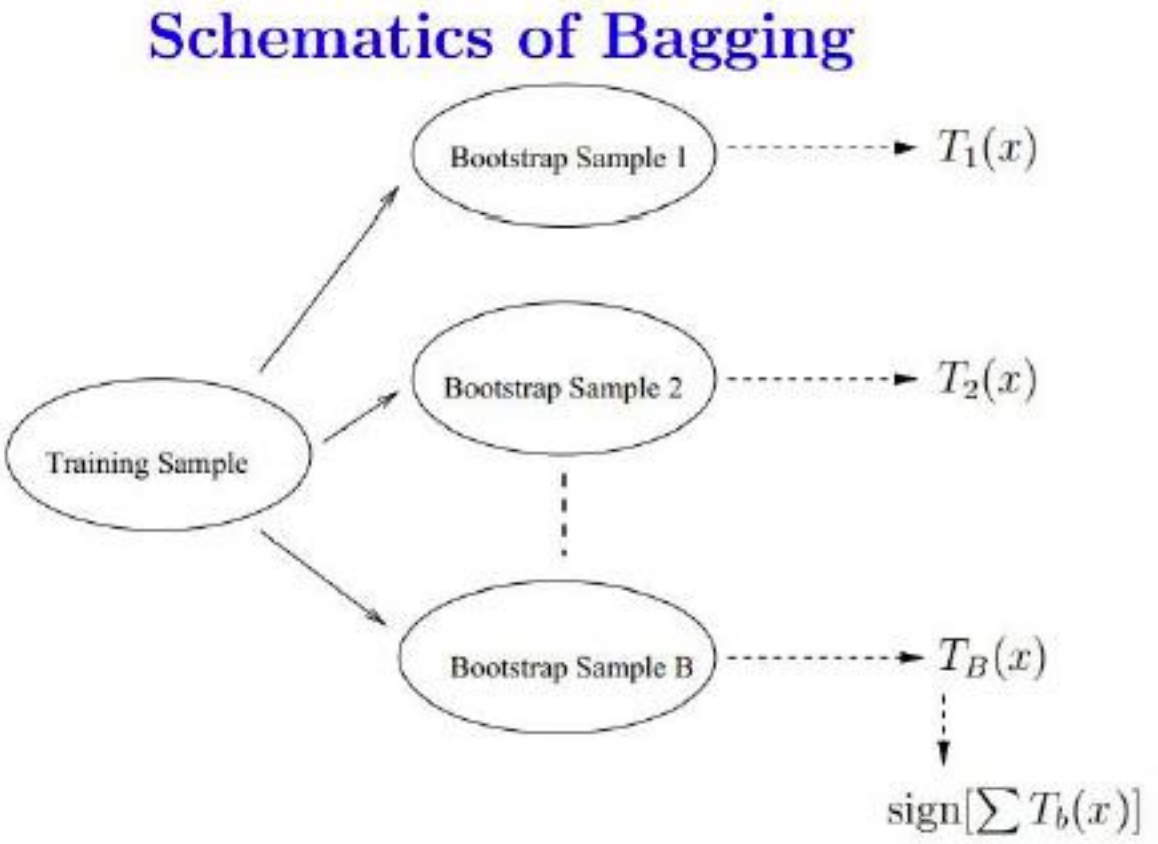
Bootstraping

- Bootstraping的名称来自成语“pull up by your own bootstraps”，意思是依靠你自己的资源，称为自助法，它是一种有放回的抽样方法。
  - 注：Bootstrap本义是指高靴子口后面的悬挂物、小环、带子，是穿靴子时用手向上拉的工具。“pull up by your own bootstraps”即“通过拉靴子让自己上升”，意思是“不可能发生的事情”。后来意思发生了转变，隐喻“不需要外界帮助，仅依靠自身力量让自己变得更好”。

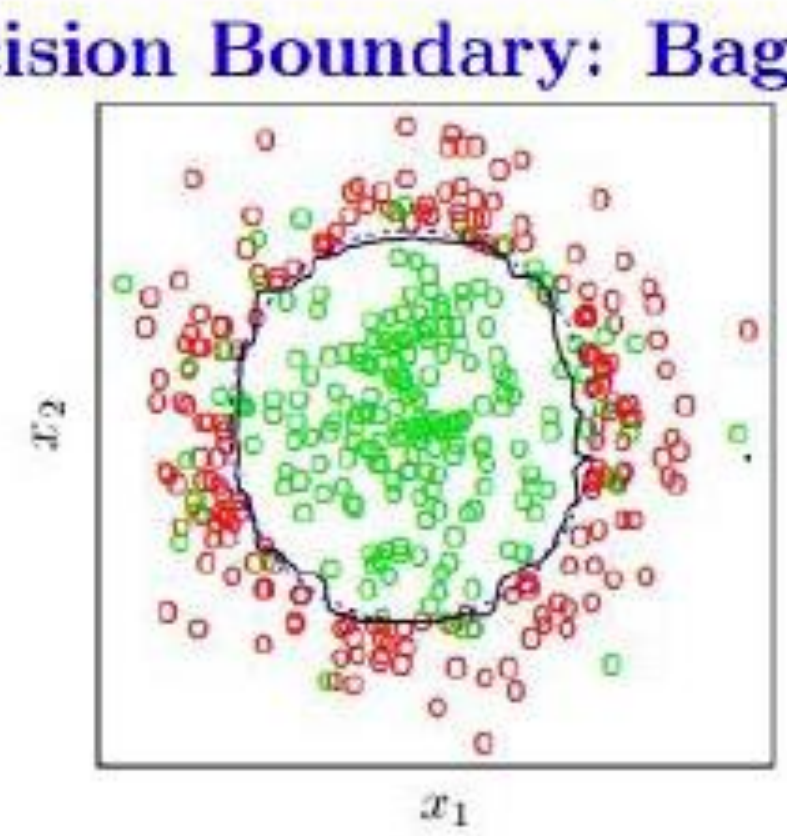
Bagging的策略

- bootstrap aggregation
- 从样本集中重采样(有重复的)选出n个样本
- 在所有属性上，对这n个样本建立分类器(ID3、C4.5、CART、SVM、Logistic回归等)
- 重复以上两步m次，即获得了m个分类器
- 将数据放在这m个分类器上，最后根据这m个分类器的投票结果，决定数据属于哪一类

Bagging


Schematics of Bagging
Bootstrap Sample 1
$T_1(x)$
Bootstrap Sample 2
$T_2(x)$
Training Sample
Bootstrap Sample B
$T_B(x)$
$\text{sign}[\sum T_b(x)]$

Bagging的结果



随机森林

- 随机森林在bagging基础上做了修改。
  - 从样本集中用Bootstrap采样选出n个样本；
  - 从所有属性中随机选择k个属性，选择最佳分割属性作为节点建立CART决策树；
  - 重复以上两步m次，即建立了m棵CART决策树
  - 这m个CART形成随机森林，通过投票表决结果，决定数据属于哪一类

随机森林/Bagging和决策树的关系

- 当然可以使用决策树作为基本分类器
- 但也可以使用SVM、Logistic回归等其他分类器，习惯上，这些分类器组成的“总分类器”，仍然叫做随机森林。

投票机制

- 简单投票机制
  - 一票否决(一致表决)
  - 少数服从多数
  - 有效多数(加权)

提升的概念

- 提升是一个机器学习技术，可以用于回归和分类问题，它每一步产生一个弱预测模型(如决策树)，并加权累加到总模型中；如果每一步的弱预测模型生成都是依据损失函数的梯度方向，则称之为梯度提升(Gradient boosting)。
- 梯度提升算法首先给定一个目标损失函数，它的定义域是所有可行的弱函数集合(基函数)；提升算法通过迭代的选择一个负梯度方向上的基函数来逐渐逼近局部极小值。这种在函数域的梯度提升观点对机器学习的很多领域有深刻影响。
- 提升的理论意义：如果一个问题存在弱分类器，则可以通过提升的办法得到强分类器。

Adaboost

- AdaBoost，是英文"Adaptive Boosting"（自适应增强）的缩写，是一种机器学习方法，由Yoav Freund和Robert Schapire提出
- AdaBoost方法的自适应在于：前一个分类器分错的样本会被用来训练下一个分类器。AdaBoost方法对于噪声数据和异常数据很敏感。
- 但在一些问题中，AdaBoost方法相对于大多数其它学习算法而言，不会很容易出现过拟合现象。

Adaboost

- 设训练数据集T={(x1,y1), (x2,y2)…(xN,yN)}
- 初始化训练数据的权值分布

$$D_1 = (w_{11}, w_{12} \cdots w_{1i} \cdots, w_{1N}), \quad w_{1i} = \frac{1}{N}, \quad i = 1,2,\cdots,N$$

Adaboost：对于m=1,2,…M

- 使用具有权值分布Dm的训练数据集学习，得到基本分类器

$$G_m(x): \chi \to \{-1,+1\}$$

- 计算Gm(x)在训练数据集上的分类误差率

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i)$$

- 计算Gm(x)的系数 $\alpha_m = \frac{1}{2}\log\frac{1-e_m}{e_m}$

Adaboost：对于m=1,2,…M

- 更新训练数据集的权值分布

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2} \cdots w_{m+1,i} \cdots, w_{m+1,N})$$

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

- 这里，Zm是规范化因子

$$Z_m = \sum_{i=1}^{N} w_{mi} \exp(-\alpha_m y_i G_m(x_i))$$

  - 它的目的仅仅是使Dm+1成为一个概率分布

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)) \Rightarrow Z_m w_{m+1,i} = w_{mi} \exp(-\alpha_m y_i G_m(x_i)) \Rightarrow Z_1 w_{2,i} = w_{1i} \exp(-\alpha_1 y_i G_1(x_i))$$

Adaboost

- 构建基本分类器的线性组合

$$f(x)=\sum_{m=1}^{M}\alpha_m G_m(x)$$

- 得到最终分类器

$$G(x)=sign(f(x))=sign\left(\sum_{m=1}^{M}\alpha_m G_m(x)\right)$$

举例

- 给定下列训练样本，试用AdaBoost算法学习一个强分类器。

序号	1	2	3	4	5	6	7	8	9	X
X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	−1	−1	−1	1	1	1	−1

解

- 初始化训练数据的权值分布

$$D_1 = (w_{11}, w_{12} \cdots w_{1i} \cdots, w_{1N}), \quad w_{1i} = \frac{1}{N}, \quad i = 1,2,\cdots,N$$

- $W_{1i} = 0.1$

m=1

- 在权值分布为D1的训练数据上，阈值v取2.5时误差率最低，故基本分类器为：

$$G_1(x)=\begin{cases}1, & x<2.5\\ -1, & x>2.5\end{cases}$$

序号	1	2	3	4	5	6	7	8	9	X
X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1

m=1

- G1(x)在训练数据集上的误差率
  e1=P(G1(xi)≠yi) =0.3
- 计算G1的系数：

$$\alpha_1 = \frac{1}{2}\log\frac{1-e_1}{e_1} = 0.4236$$

- f1(x)=0.4236*G1(x)
- 分类器sign(f1(x))在训练数据集上有3个误分类点。

m=1

- 更新训练数据的权值分布：

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2} \cdots w_{m+1,i} \cdots, w_{m+1,N}),$$

$$w_{m+i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

- D2=(0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.0715,0.1666, 0.1666, 0.1666, 0.0715)
  - 计算D2，是为下一个基本分类器使用
- f1(x)=0.4236*G1(x)
- 分类器sign(f1(x))在训练数据集上有3个误分类点。

m=2

- 在权值分布为D2的训练数据上，阈值v取8.5时误差率最低，故基本分类器为：

$$G_2(x)=\begin{cases}1, & x<8.5\\ -1, & x>8.5\end{cases}$$

X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	−1	−1	−1	1	1	1	−1
w	0.0715	0.0715	0.0715	0.0715	0.0715	0.0715	0.1666	0.1666	0.1666	0.0715

m=2

- G2(x)在训练数据集上的误差率

  e2=P(G2(xi)≠yi) = 0.2143(0.0715*3)

- 计算G2的系数：

$$\alpha_2 = \frac{1}{2}\log\frac{1-e_2}{e_2} = 0.6496$$

m=2

- 更新训练数据的权值分布：

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2} \cdots w_{m+1,i} \cdots, w_{m+1,N})$$

$$w_{m+i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

- D3=(0.0455, 0.0455, 0.0455, 0.1667, 0.1667, 0.01667,0.1060, 0.1060, 0.1060, 0.0455)
- f2(x)=0.4236G1(x) + 0.6496G2(x)
- 分类器sign(f2(x))在训练数据集上有3个误分类点

m=3

- 在权值分布为D3的训练数据上，阈值v取5.5时误差率最低，故基本分类器为：

$$G_3(x)=\begin{cases}1, & x>5.5\\ -1, & x<5.5\end{cases}$$

X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1
w	0.0455	0.0455	0.0455	0.1667	0.1667	0.1667	0.1060	0.1060	0.1060	0.0455

m=3

- G3(x)在训练数据集上的误差率

  e3=P(G3(xi)≠yi) = 0.1820(0.0455*4)

- 计算G3的系数：

$$\alpha_3 = \frac{1}{2}\log\frac{1-e_3}{e_3} = 0.7514$$

m=3

- 更新训练数据的权值分布：

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2} \cdots w_{m+1,i} \cdots, w_{m+1,N}),$$

$$w_{m+i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

- D4=(0.125, 0.125, 0.125, 0.102, 0.102, 0.102, 0.065,0.065, 0.065, 0.125)
- f3(x)=0.4236G1(x) + 0.6496G2(x)+0.7514G3(x)
- 分类器sign(f3(x))在训练数据集上有0个误分类点

感谢大家！

恳请大家批评指正！

附：GBDT

- 给定输入向量x和输出变量y组成的若干训练样本(x1,y1),(x2,y2),…,(xn,yn)，目标是找到近似函数F(x)，使得损失函数L(y,F(x))的损失值最小
- L(y,F(x))的典型定义为 $L(y,F(\vec{x}))=\frac{1}{2}(y-F(\vec{x}))^2$

$$L(y,F(\vec{x}))=|y-F(\vec{x})|$$

- 假定最优函数为：$F^*(\vec{x})=\arg\min_F E_{(x,y)}[L(y,F(\vec{x}))]$
- 假定F(x)是一族基函数fi(x)的加权和

$$F(\vec{x})=\sum_{i=1}^{M}\gamma_i f_i(x)+const$$

提升算法推导

- 梯度提升方法寻找最优解F(x)，使得损失函数在训练集上的期望最小。方法如下：
- 首先，给定常函数F0(x)：

$$F_0(\vec{x}) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

- 以贪心的思路扩展得到Fm(x)：

$$F_m(\vec{x}) = F_{m-1}(\vec{x}) + \arg\min_{f \in H} \sum_{i=1}^{n} L(y_i, F_{m-1}(\vec{x}_i) + f(\vec{x}_i))$$

梯度近似

- 贪心法在每次选择最优基函数f时仍然困难
  - 使用梯度下降的方法近似计算
  - 将样本带入基函数f得到f(x1),f(x2),...,f(xn)，从而L退化为向量L(y1,f(x1)),L(y2,f(x2)),...,L(yn,f(xn))

$$F_m(\vec{x}) = F_{m-1}(\vec{x}) - \gamma_m \sum_{i=1}^{n} \nabla_f L(y_i, F_{m-1}(\vec{x}_i))$$

- 上式中的权值γ为梯度下降的步长，使用线性搜索求最优步长：

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(\vec{x}_i) - \gamma \cdot \nabla_f L(y_i, F_{m-1}(\vec{x}_i)))$$

提升算法

- 初始给定模型为常数 $F_0(\vec{x}) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$
- 对于m=1到M
  - 计算伪残差 $r_{im} = \left[\frac{\partial L(y_i, F(\vec{x}_i))}{\partial F(\vec{x}_i)} \right]_{F(\vec{x})=F_{m-1}(\vec{x})} \quad i = 1,2,\cdots,n$
  - 使用数据 计算拟合残差的基函数$f_m(x)$
  - 计算步长 $\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(\vec{x}_i) - \gamma \cdot f_m(\vec{x}_i))$
    - 一维优化
  - 更新模型 $F_m(\vec{x}) = F_{m-1}(\vec{x}) - \gamma_m f_m(\vec{x}_i)$

梯度提升决策树GBDT

- 梯度提升的典型基函数即决策树(尤其是CART)
- 在第m步的梯度提升是根据伪残差数据计算决策树tm(x)。令树tm(x)的叶节点数目为J，即树tm(x)将输入空间划分为J个不相交区域R1m,R2m,...,RJm，并且决策树tm(x)可以在每个区域中给出某个类型的确定性预测。使用指示记号I(x)，对于输入x， tm(x) 为：

$$t_m(\vec{x}) = \sum_{j=1}^{J} b_{jm} I(\vec{x} \in R_{jm})$$

- 其中，b_{jm}是样本x在区域R_{jm}的预测值。

GDBT

- 使用线性搜索计算学习率，最小化损失函数

$$F_m(\vec{x}) = F_{m-1}(\vec{x}) + \gamma_m \cdot t_m(\vec{x}_i)$$

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(\vec{x}_i) + \gamma \cdot t_m(\vec{x}_i))$$

- 进一步：对树的每个区域分别计算步长，从而系数b_{jm}被合并到步长中，从而：

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

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{\vec{x}_i \in R_{jm}} L(y_i, F_{m-1}(\vec{x}_i) + \gamma \cdot t_m(\vec{x}_i))$$

参数设置和正则化

- 对训练集拟合过高会降低模型的泛化能力，需要使用正则化技术来降低过拟合。
  - 对复杂模型增加惩罚项，如：模型复杂度正比于叶结点数目或者叶结点预测值的平方和等。
  - 用于决策树剪枝
- 叶结点数目控制了树的层数，一般选择$4 \leq J \leq 8$。
- 叶结点包含的最少样本数目
  - 防止出现过小的叶结点，降低预测方差
- 梯度提升迭代次数M：
  - 增加M可降低训练集的损失值，但有过拟合风险
  - 交叉验证

衰减因子、降采样

- 衰减Shrinkage $F_m(\vec{x}) = F_{m-1}(\vec{x}) + \nu \cdot \gamma_m f_m(\vec{x}_i), \quad 0 < \nu \leq 1$
  - 称ν为学习率
  - ν=1即为原始模型；推荐选择ν<0.1的小学习率。过小的学习率会造成计算次数增多。
- 随机梯度提升Stochastic gradient boosting
  - 每次迭代都对伪残差样本采用无放回的降采样，用部分样本训练基函数的参数。令训练样本数占所有伪残差样本的比例为f；f=1即为原始模型；推荐0.5≤f≤0.8。
  - 较小的f能够增强随机性，防止过拟合，并且收敛的快。
  - 降采样的额外好处是能够使用剩余样本做模型验证。

GBDT总结

- 函数估计本来被认为是在函数空间而非参数空间的数值优化问题，而阶段性的加性扩展和梯度下降手段将函数估计转换成参数估计。
- 损失函数是最小平方误差、绝对值误差等，则为回归问题；而误差函数换成多类别Logistic似然函数，则成为分类问题。
- 对目标函数分解成若干基函数的加权和，是常见的技术手段：神经网络、径向基函数、傅立叶/小波变换、SVM都可以看到它的影子。