决策树

https://blog.csdn.net/weixin_36586536/article/details/80468426

信息熵(information entropy)

在信息论和概率统计中，熵(entropy)是表示随机变量不确定性的度量，设X是一个取有限值的离散随机变量，其概率分布为

$$P(X=x_i)=p_i, i=1,2,...,n$$

则随机变量X的熵定义为

$$H(X)=-\sum_{i=1}^{n} p_i \log p_i$$

条件熵

设有随机变量(X,Y)(X,Y)。条件熵H(Y|X)H(Y|X)表示在已知随机变量XX的条件下随机变量YY的不确定性。随机变量XX给定的条件下随机变量YY的条件熵H(Y|X)H(Y|X)定义为XX给定条件下YY的条件概率分布的熵对XX的数学期望

$$H(YX)=\sum_{i=1}^{n} p_i H(Y|X=x_i)$$

$p_i=P(X=x_i), i=1,2,...,n$

信息增益(information gain) - ID3

信息增益表示得知特征XX的信息而使得类YY的信息的不确定性减少程度。接下来给出定义，特征AA对训练数据集DD的信息增益g(D,A)g(D,A),为集合DD的熵H(D)H(D)与特征AA给定条件下DD的条件熵H(D|A)H(D|A)之差，即

$$g(D,A)=H(D)-H(D|A)$$

(1) 计算数据集D的熵H(D)

$$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|}$$

(2) 计算特征A对训练数据集D的条件熵H(D|A)

$$H(D|A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log_2\frac{|D_{ik}|}{|D_i|}$$

|D|为样本容量，设有K个类C_k, |C_k|为属于类C_k的样本个数。设特征A有n个不同取值，根据特征A的取值将D划分为n个子集D1,D2,...,Dn, D_{ik}为子集Di中属于类C_k的集合。

信息增益率(information gain ratio) - C4.5

特征A对训练数据集D的信息增益g_R(D,A)定义为其信息增益g(D,A)与训练数据集D关于特征A的值的熵$H_A(D)$之比，即

$$g_R(D,A)=\frac{g(D,A)}{H_A(D)}$$

回归树(CART)

基尼指数(Gini index)

分类问题中，假设有K个类，样本点属于第k类的概率为p_k,则概率分布的基尼指数定义为

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

则在特征A的条件下,集合D的基尼指数定义为

$$Gini(D,A=a)=\frac{|D_1|}{|D|}Gini(D_1)+\frac{|D_2|}{|D|}Gini(D_2)$$

集成方法(Ensemble Method)

boosting

一系列个体学习，之间有比较强的依赖关系，会降偏差
算法: AdaBoost, GBDT(回归树), xgBoost, lightGBM

bagging(bootstrap aggregating)

并行执行，分类问题投票，回归问题均值解决，会降低方差
算法: Random forest

AdaBoost

https://blog.csdn.net/v_JULY_v/article/details/40718799

整个Adaboost 迭代算法就3步:
(1)初始化训练数据的权值分布。如果有N个样本，则每一个训练样本最开始时都被赋予相同的权值：1/N。

$$D_i = (w_{11}, w_{12}, ..., w_{1i}, ..., w_{1N}), w_{1i} = \frac{1}{N}, i = 1, 2, ..., N$$

(2)训练弱分类器。具体训练过程中，如果某个样本点已经被准确地分类，那么在构造下一个训练集中，它的权值就被降低；相反，如果某个样本点没有被准确地分类，那么它的权值就得到提高。然后，权值更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。

基本分类器:

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

误差:

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

其中m = 1,2,..,M，代表第m轮迭代。i代表第i个样本。w 是样本权重。I指示函数取值为1或0，当I指示函数括号中的表达式为真时，I 函数结果为1；当I函数括号中的表达式为假时，I 函数结果为0。

由上述式子可知，Gm(x)在训练数据集上的误差率em就是被Gm(x)误分类样本的权值之和。

计算最优弱分类器的权重:

$$\alpha_m = \frac{1}{2}\log(\frac{1-e_m}{e_m})$$

$$w_{m+1,i} = \frac{w_{mi}}{z_m}\exp(-\alpha_m y_i G_m(x_i))$$

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

其中α是弱分类器的权重。当样本被正确分类时，y 和 Gm 取值一致，则新样本权重变小；当样本被错误分类时，y 和 Gm 取值不一致，则新样本权重变大

(3)将各个训练得到的弱分类器组合成强分类器。各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器的权重，使其在最终的分类函数中起着较小的决定作用。换言之，误差率低的弱分类器在最终分类器中占的权重较大，否则较小。

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

GBDT (Gradient Boosting Decison Tree)

http://wepon.me/files/gbdt.pdf
https://blog.csdn.net/zpalyq110/article/details/79527653
https://www.jianshu.com/p/a72539acafe5

XGBoost

https://www.jianshu.com/p/ac1c12f3fba1
https://www.jianshu.com/p/a72539acafe5

GBDT算法可以看成是由K棵树组成的加法模型:

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

其中为F所有树组成的函数空间

目标函数定义为:

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

其中Ω表示决策树的复杂度，比如，可以考虑树的节点数量、树的深度或者叶子节点所对应的分数的L2范数等等

具体地，我们从一个常量预测开始，每次学习一个新的函数，过程如下:

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

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

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

...

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

目标函数变换为[1]:

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

举例说明，假设损失函数为平方损失(square loss)，则目标函数为[2]::

$Obj^{(t)} = \sum_{i=1}^{n}(y_i - (\hat{y}_i^{t-1} + f_t^2(x_i))) + \Omega(f_t) + Constant$

$Obj^{(t)} = \sum_{i=1}^{n}[y_i^2 - 2y_i(\hat{y}_i^{t-1} + f_t(x_i)) + (\hat{y}_i^{t-1} + f_t^2(x_i))] + \Omega(f_t) + Constant$

$Obj^{(t)} = \sum_{i=1}^{n}[y_i^2 - 2y_i\hat{y}_i^{t-1} - 2y_i f_t(x_i) + (\hat{y}_i^{t-1})^2 + 2(\hat{y}_i^{t-1})f_t(x_i) + f_t^2(x_i)] + \Omega(f_t) + Constant$

$Obj^{(t)} = \sum_{i=1}^{n}[(\hat{y}^{t-1} - y_i)^2 + 2(\hat{y}^{t-1} - y_i)f_t(x_i) + f_t^2(x_i)] + \Omega(f_t) + Constant$

其中$(\hat{y}^{t-1} - y_i)$为残差，因此，使用平方损失函数时，GBDT算法的每一步在生成决策树时只需要拟合前面的模型的残差。

根据泰勒公式把函数f(x + \Delta x)在点处二阶展开，可得到如下等式[3]:

$$f(x + \Delta x) \approx f(x) + f^{'}(x)\Delta x + \frac{1}{2}f^{''}(x)\Delta x^2$$

目标函数变换为[4]:

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

由于函数中的常量在函数最小化的过程中不起作用，因此我们可以从等式(4)中移除掉常量项，得[5]：

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

决策树的复杂度可以由正则项:

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

一颗生成好的决策树，假设其叶子节点个数为T, 该决策树是由所有叶子节点对应的值组成的向量$\omega \in R^T$, 以及一个把特征向量映射到叶子节点索引（Index）的函数q, 因此，策树可以定义为

$$f_t(x) = \omega_{q(x)}$$

目标函数变换为[6]:

$Obj^{(t)} \approx \sum_{i=1}^{n}[g_i\omega_{q(x)} + \frac{1}{2}h_i\omega_{q(x)}^2] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T}\omega_j^2$

$Obj^{(t)} \approx \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T}\omega_j^2$

$Obj^{(t)} \approx \sum_{i=1}^{n}[(\sum_{i\in I_j} g_i)\omega_{q(x)} + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)\omega_{q(x)}^2] + \gamma T$

定义$G_j = \sum_{i\in I_j} g_i$, $H_j = \sum_{i\in I_j} h_i$

$Obj^{(t)} \approx \sum_{j=1}^{T}[G_i\omega_j + \frac{1}{2}(H_i + \lambda)\omega_j^2] + \gamma T$

假设树的结构是固定的，即函数q(x)确定，令函数$Obj^{(t)}$的一阶导数等于0，即可求得叶子节点对应的值为[7]::

$$\omega_j^* = -\frac{G_j}{H_j + \lambda}$$

目标函数变换为[8]:

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

综上，为了便于理解，单颗决策树的学习过程可以大致描述为:

1.枚举所有可能的树结构q

2.用等式(8)为每个q计算其对应的分数Obj，分数越小说明对应的树结构越好

3.根据上一步的结果，找到最佳的树结构，用等式(7)为树的每个叶子节点计算预测值

树分裂原则:

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

带入[8]

假设未分裂的情况下的损失值:

$$\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}$$

假设分裂的情况下左右叶子节点的损失值:

$$\frac{G_L^2}{H_L+\lambda}, \frac{G_R^2}{H_R+\lambda}$$

所以如果需要分裂，求取下面的最大值:

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

γ可以用来控制树的复杂度，进一步来说，利用γ来作为阈值，只有大于γ时候才选择分裂。这个其实起到预剪枝的作用