决策树

Decision Tree

信息论基础

- 信息是对不确定性的消除。
- 自信息：$I(a_i) = -\log P(a_i)$
- 性质：设a_1, a_2为两个随机事件

(1) 若$P(a_1) \geq P(a_2)$, 则$I(a_1) \leq I(a_2)$

(2) 若$P(a_1) = 1$，则$I(a_1) = 0$

(3) 若$P(a_1) = 0$，则$I(a_1) = \infty$

(4) 若a_1, a_2为独立事件，则$I(a_1, a_2) = I(a_1) + I(a_2)$.

信息熵是指随机系统的总体信息量，用所有随机事件自信息的统计平均来表示:

$$H(X) = f(p_1, p_2, \ldots, p_N) = -\sum_{x \in X} p(x) log(x)$$

- 性质:

1、对称性: $f(p_1, p_2, \ldots, p_N) = f(p_{k(1)}, p_{k(2)}, \ldots, p_{k(N)})$

2、非负性: $H(X) \geq 0$

3、可加性: $H(X, Y) = H(X) + H(Y|X)$

4、条件减少熵: $H(X|Y) \leq H(X)$

5、最大离散熵定理: $f(p_1, p_2, \ldots, p_N) \leq f(\frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N}) = \log N = \log |X|$

联合熵：一对离散随机变量(X, Y)的联合熵定义为：

$$H(X,Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log p(x,y)$$

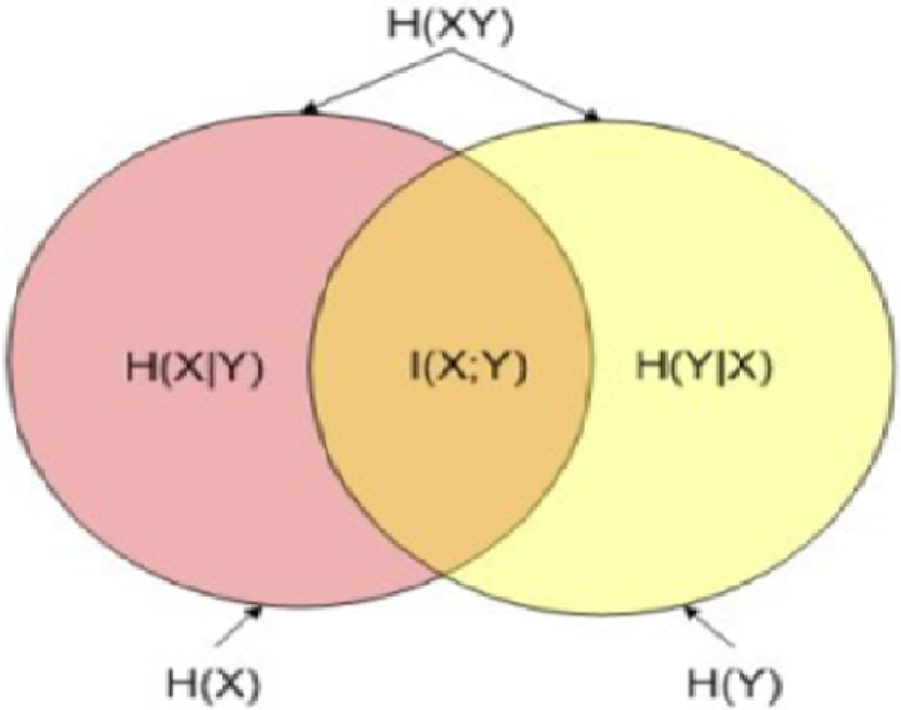
条件熵：$H(Y|X) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log p(y|x)$

$H(Y|X) = \sum_{x \in \mathcal{X}} p(x) H(Y|X = x)$

- 条件熵是不对称的。

互信息：$I(X;Y) = H(X) - H(X|Y)$ 或

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



I(X;Y)=H(X)-H(X|Y)

I(X;Y)=H(Y)-H(Y|X)

I(X;Y)=H(X)+H(Y)-H(XY)

- 互信息是对称的。

相对熵（KL散度）：

设$p(x)$、$q(x)$是关于随机变量X的两个概率分布，则p相对于q的相对熵是：

$$D_{KL}(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

性质：1、如果$p(x)$和$q(x)$两个分布相同，当且仅当相对熵等于0

2、 $D_{KL}(p\|q) \neq D_{KL}(q\|p)$

3、$D_{KL}(p\|q) \geq 0$

- 定理： 互信息是联合分布和两个边缘分布之间的KL散度

$$I(X,Y) = KL(p(x,y)||p(x)p(y))$$

因此互信息是非负的。

交叉熵（Cross entropy）：

设$p(x)$、$q(x)$是关于随机变量X的两个概率分布，使用分布$q(x)$表示目标分布$p(x)$的困难程度：

$$H(p,q) = -\sum_i p(x_i)\log q(x_i)$$

性质：1、$D_{KL}(p,q) = H(p,q) - H(p)$

- 熵：可以表示一个事件A的自信息量，也就是A包含多少信息。
- KL散度：可以用来表示从事件A的角度来看，事件B有多大不同。
- 交叉熵：可以用来表示从事件A的角度来看，如何描述事件B.使用分布$q(x)$表示目标分布$p(x)$的困难程度

在机器学习中，（1）希望学到的模型的分布和真实分布一致：

$$P(model) \simeq P(real)$$

（2）但是真实分布不可知，假设训练数据时从真实数据中独立同分布采样的：

$$P(train) \simeq P(real)$$

（3） 因此，我们希望学到的模型分布至少和训练数据的分布一致：

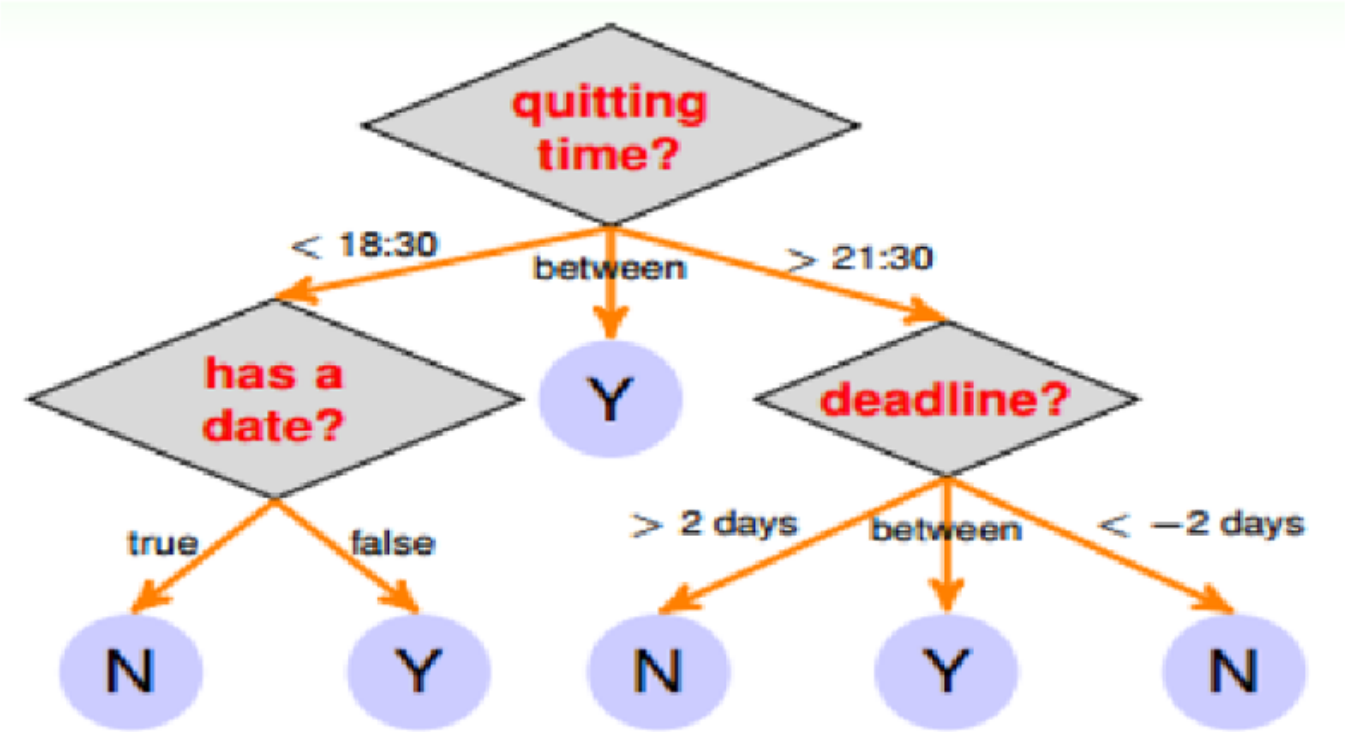
$$P(train) \simeq P(model)$$

交叉熵可以用来计算学习模型分布与训练分布之间的差异"

$$\begin{aligned} & L(Y, P(Y|\mathbf{X})) \\ = \quad & -\log P(Y|\mathbf{X}) \\ = \quad & -\frac{1}{N}\sum_{i=1}^{N}(1-y_i)\log(1-p(\mathbf{x}_i)) - y_i \log p(\mathbf{x}_i) \end{aligned}$$

决策树

决策树模型是一种传统的算法，与人类思维十分相似



- 决策树的基本结构：一颗决策树(decision tree) 包括若干内部结点(对应于属性测试) 和若干叶结点(对应于决策结果).
  - 根结点包含训练集全集, 每个内部结点包含的样本集合根据属性测试的结果被划分到子结点中用以进一步的属性测试.
  - 从根结点到每个叶结点的路径对应了一个判定测试序列.
  - 决策树设计的关键是如何设计属性测试方法, 或属性划分准则.
- $G(x) = \sum_{t=1}^{T} [\![q_t(x)]\!] \cdot g_t(x)$
  - $g_t(x)$表示树的叶子
  - $q_t(x)$表示边上的条件
- $G(x) = \sum_{c=1}^{C} [\![b(x) = c]\!] \cdot G_c(x)$
  $G_c(x)$表示第c个分支下的子树
  $b(x) = c$表示第c个分支的条件

决策树可以分为两部分：根和子树

- 优点
  - 模型直观，便于理解，应用广泛
  - 算法简单，容易实现
  - 训练和预测时，效率较高
- 缺点
  - 缺少足够的理论支持
  - 如何选择合适的树结构对初学者来说比较困惑
  - 决策树代表性的演算法比较少

Function DecisionTree（data $D = \{(x_n, y_n)\}_{n=1}^{N}$）

If 终止条件 满足

Return 基本算法$g_t(x)$

Else (1) 学习分支条件$b(x)$

(2) 将D分成C个部分$D_c = \{(x_n, y_n) : b(x_n) = c\}$

(3) 构造子树 $G_c = DecisionTree(D_c)$

(4) Return $G(x) = \sum_{c=1}^{C} [\![b(x) = c]\!] \cdot G_c(x)$

- 决策树生成过程的停止条件
  - 当前结点包含的样本全属于同一类别, 无需划分.
  - 当前结点包含的属性集为空, 或是当前结点包含的样本在所有属性上取值相同, 无法划分.
  - 当前结点包含的样本集合为空, 不能划分.
- 划分选择：我们希望决策树的分支结点所包含的样本尽可能属于同一类别, 即结点的纯度(purity) 越来越高.
  - 信息增益
  - 增益率
  - 基尼系数

1. 信息增益(information gain)

(1). 计算样本集合D中第c类样本所占的比例:

$$p_c := \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(y_i = c)$$

样本集合D的信息熵为: $H(D) := -\sum_{c=1}^{C} p_c \log p_c$

(2). 假设某属性有K个可能的取值$a_1, a_2, \ldots, a_K$, 若使用该属性来对样本集D进行划分, 则会产生K个分支结点, 我们记D_{a_k}为所有在属性上取值为a_k的样本,计算D_{a_k}的信息熵$H(D_{a_k})$.

(3). 考虑到不同的分支结点所包含的样本数不同, 给分支结点赋予权重$\frac{|D_{a_k}|}{|D|}$, 计算出用该属性对样本集D进行划分所获得的信息增益

$$g(D, A) = H(D) - \sum_{k=1}^{K}\frac{|D_{a_k}|}{|D|}H(D_{a_k})$$

2.增益率(gain ratio)

$$g_R(D,a) = \frac{g(D,a)}{H_a(D)}$$

- 信息增益对可取值数较多的属性有所偏好
- 增益率对可取值数较少的属性有所偏好

3.基尼系数

(1).计算样本集合D中第c类样本所占的比例p_c, 以及样本集合D的基尼值

$$\text{gini}(D) := \sum_{c=1}^{C}\sum_{c'\neq c} p_c p_{c'} = \sum_{c=1}^{C} p_c(1-p_c) = 1 - \sum_{c=1}^{C} p_c^2$$

(2).计算每个分支D_{a_k}的基尼值$\text{gini}(D_{a_k})$

(3).用该属性对样本集D进行划分所对应的基尼指数是

$GI := \sum_{k=1}^{K} \frac{|D_{a_k}|}{|D|}\text{gini}(D_{a_k})$

- 二分法（bi-partition）
  - 假设某属性在D上出现了K个不同的值，将这些值按从小到大进行排序，记为$a_1, a_2, \ldots, a_K$. 基于划分点t可将D分为子集D_t^-和D_t^+
  - 考察$K-1$个候选划分点$\{\frac{a_k+a_{k+1}}{2} | 1 \leq k < K\}$
  - 可以像离散属性值那样考虑信息增益：

$$g(D,a) := \max_t H(D) - \frac{|D_t^+|}{|D|} H(D_t^+) - \frac{|D_t^-|}{|D|} H(D_t^-)$$

学习算法	属性划分准则
ID3 (iterative dichotomiser) [11]	信息增益
C4.5 [12]	信息增益 + 增益率，二分法
CART (classification and regression tree) [2]	基尼指数

- 剪枝：是决策树学习算法应对过拟合的主要手段.
  - 预剪枝：在决策树生成过程中, 对每个结点在划分前进行估计,若当前结点的划分不能带来决策树泛化性能的提升(如验证集性能提升), 则停止划分并将当前结点标记为叶结点.
    - 利：降低过拟合，显著减少训练时间和测试时间开销
    - 弊：基于贪心本质禁止分支展开，有欠拟合的风险
  - 后剪枝：先从训练集生成一棵完整的决策树, 然后自底向上地对非叶结点进行考察, 若将该结点对应的子树替换为叶结点能带来决策树泛化性能的提升或保持不变, 则将该子树替换为叶结点.
    - 利：欠拟合风险很小，泛化能力往往优于预剪枝
    - 弊：训练时间开销比未剪枝决策树和预剪枝决策树都要大得多.

- 缺失值处理

1.属性值缺失的情况下进行划分属性选择?
a.给定训练集D和某属性a,$\tilde{D}_a$为D中在a上没有缺失值的样本子集.
b.为每个样本赋予权重w_i,无缺失值样本$\tilde{D}_a$中第c类所占比例为:

$$\tilde{p}_{ac} := \frac{\sum_{i=1}^{N} \mathbb{I}((\mathbf{x}_i, y_i) \in \tilde{D}_a \wedge y_i = c) w_i}{\sum_{i=1}^{N} \mathbb{I}((\mathbf{x}_i, y_i) \in \tilde{D}_a) w_i}$$

c.无缺失值样本$\tilde{D}_a$的信息熵定义为: $H(\tilde{D}_a) := -\sum_{c=1}^{C} \tilde{p}_{ac} \log \tilde{p}_{ac}$

d.无缺失值样本所占的比例定义为:

$$\rho_a := \frac{\sum_{i=1}^{N} \mathbb{I}((\mathbf{x}_i, y_i) \in \tilde{D}_a) w_i}{\sum_{i=1}^{N} \mathbb{I}((\mathbf{x}_i, y_i) \in D) w_i}$$

e.无缺失值样本$\tilde{D}_a$中在属性a上取值为a_k的样本所占的比例为:

$$r_{a_k} := \frac{\sum_{i=1}^{N} \mathbb{I}((\mathbf{x}_i, y_i) \in \tilde{D}_a \wedge x_j = a_k) w_i}{\sum_{i=1}^{N} \mathbb{I}((\mathbf{x}_i, y_i) \in \tilde{D}_a) w_i}$$

f. 则信息增益的计算公式为:

$$G(D, a) := \rho_a (H(\tilde{D}_a) - \sum_{k=1}^{K} r_{a_k} H(\tilde{D}_{a_k}))$$

2.给定划分属性, 若样本在该属性上的值缺失,如何对样本进行划分?

- 若样本x在划分属性a上的取值已知, 则将x划入与其取值对应的子结点, 且样本权值在子结点中保持为w_i.
- 若样本x在划分属性a上的取值未知, 则将x同时划入所有子结点, 且样本权值在与属性a_k对应的子结点中调整为$r_{a_k} w_i$.

随机森林

Bagging算法会减少不同g_t的方差

Decision Tree会增大不同g_t的方差
随机森林使用Bagging的方式把众多的Decision Tree进行uniform结合起来

Function RandomForest(D)

For $t = 1, 2, \cdots, T$

(1)获得D_t=$bootstrapping(D)$

(2)得到g_t="DecisionTree" (D_t)

Return G=$Uniform(\{g_t\})$

- 不同决策树可以由不同主机并行训练生成，效率很高
- 随机森林算法继承了C&RT的优点
- 将所有的决策树通过bagging的形式结合起来，避免了单个决策树造成过拟合的问题。

增强的随机森林

- 随机特征抽取：随机选取d'个特征, $\phi(x) = (x_{i_1}, x_{i_2}, \cdots, x_{i_d})$，然后构建决策树
- 特征线性组合：$\phi_i(x) = p_i^T x$, i表示不同的分支

特征选择：从d维特征到d'维特征的子集变换$\phi(x)$称为特征选择

- 冗余特征、不相关的特征

特征选择的优点

- 提高效率，特征越少，模型越简单
- 正则化，防止特征过多出现过拟合
- 去除无关特征，保留相关性大的特征，解释性强

特征选择的缺点

- 筛选特征的计算量较大
- 不同特征组合，也容易发生过拟合
- 容易选到无关特征，解释性差

特征重要性

线性模型：$score = w^T x = \sum_{i=1}^{d} w_i x_i$

特征i的重要性定义为：$importance(i) = |w_i|$

随机森林中的特征选择：Random test

- 采用uniform或者gaussian抽取随机值替换原特征
- 随机排序测试：通过Permutation的方式将原来的所有N个样本的第i个特征值重新打乱分布

AdaBoost-DTree

Function AdaBoost-DTree(D)

For $t = 1, 2, \cdots, T$

(1)构造样本权重u^t

(2)得到$g_t = \text{"}DTree\text{"}(D, u^t)$

(3)计算子分类器权重$\alpha^t = ln\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$

Return $G(x) = sign(\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}))$

Gradient Boosting Decision Tree (GBDT)

$s_1 = s_2 = \cdots = s_N = 0$

For $t = 1, 2, \cdots, T$

(1)利用决策树算法获得$g_t = DecisionTree((\mathbf{x}_n, y_n - s_t(\mathbf{x}_n)))$
(2)利用线性回归算法计算$\alpha_t = LR((g_t(x_n), y_n - s_t(\mathbf{x}_n)))$

(3)更新$s_t(\mathbf{x}_n) = s_t(\mathbf{x}_n) + \alpha_t g_t(\mathbf{x}_n)$

返回 $G(x) = sign(\sum_{t=1}^{T} \alpha_t g_t(x))$