决策树

决策树(decision tree)是一种基本的分类与回归算法，呈树形结构。它可以认为是if-then规则的集合，也可以认为是定义在特征空间与类空间上的条件概率分布，主要优点是具有可读性、分类速度快。学习时，利用训练数据，根据损失函数最小化的原则建立决策树模型，预测时，对新的数据，进行分类。决策树学习有三个步骤：特征选择、生成和修剪。分别介绍ID3、C4.5和CART算法。

决策树模型与学习

决策树模型

分类决策树模型是一种描述对实例进行分类的树形结构，由结点(node)和有向边(directed edge)组成。结点有两种：内部结点和叶结点。内部结点表示一个特征或属性，叶结点表示一个类。

用决策树分类，从根结点开始，对实例的某一特征进行测试，根据测试结果，将实例分配到其子结点；这时，一个子结点对应着该特征的一个取值。如此递归下去，直到叶结点，将该实例分到叶结点的类中。

决策树与if-then规则

决策树还可以表示给定特征条件下类的条件概率分布，这一条件概率分布定义在特征空间的一个划分(partition)上，将特征空间划分为互不相交的单元或区域，并在每个单元定义一个类的概率分布就构成了一个条件概率分布。决策树的一条路径对应于划分中的一个单元。决策树所表示的条件概率分布由各个单元给定条件下类的条件概率分布组成。假设X为表示特征的随机变量，Y为类的随机变量，那么这个条件概率分布可以表示为$P(Y|X)$。X取值于给定划分下单元的集合，Y取值于类的集合。各叶结点上的条件概率往往偏向某一个类，即属于某一类的概率较大。决策树分类时将该结点的实例强行分到条件概率大的那一类去。

决策树学习

决策树学习，给定训练集

$$D=\{(x^1,y^1),\cdots,(x^N,y^N)\}$$

且特征数为n，类数为K，N为样本数。学习的目标是由训练集建立一个决策树模型。

决策树学习本质上是从训练集中归纳出一组分类规则，这可能有多个，也可能一个没有，我们需要的是找到于训练集矛盾较小的决策树，同时有很好的泛化能力。

决策树学习用损失函数表达这一目标，如下所述，决策树学习的损失函数通常是正则化的极大似然函数。当损失函数确定以后，学习问题就变为在损失函数意义下选择最优决策树的问题。因为从所有决策树中选择最优是NP完全问题，所以现实中决策树学习算法通常采用**启发式方法**，**近似**求解这一最优化问题。这样得到的决策树是次最优(sub-optimal)的。

决策树学习的算法通常是一个递归地选择最优特征，并根据该特征对训练集进行分割，使得对各个子数据集有一个最好的分类的过程，这一过程对应着对特征空间的划分，也对应着决策树的构建。

开始，构建根结点，将所有训练数据都放在根结点中；选择一个最优特征，按照这一特征将训练集分割成子集，使各个子集有一个在当前条件下的最好分类。如果这些子集已经能够被基本正确分类，那么构建叶节点，并将这些子集分到对应的叶结点中；如果还有子集不能正确分类，那么对这些子集选择新的最优特征，继续分割。如此递归下去，直到所有训练子集都被基本正确分类或者没有合适的特征。最后每个子集都被分到叶结点上，这就形成了决策树。

以上方法生成的决策树不一定有很好的泛化能力，即可能过拟合。我们需要对已生成的树自下而上进行剪枝，将树变得简单。具体地，就是去掉过于细分的叶结点，将其回退到父结点，甚至更高的结点，将这些结点改为新的叶结点。

如果特征数量过多，也可以在开始的时候只留下有用的特征。

可以看出，决策树的学习算法包含**特征选择、决策树的生成与决策树的剪枝**过程。由于决策树表示一个条件概率分布，所以深浅不同的决策树对应着不同复杂度的概率模型。决策树的生成对应于模型的局部选择，剪枝对应于模型的全局选择。生成只考虑局部最优，剪枝则考虑全局最优。

学习算法常用的有ID3、C4.5和CART。下面结合这些算法来讨论。

特征选择

特征选择问题

特征选择在于选取对训练数据具有分类能力的特征，如果一个特征对分类的结果没有帮助，那么扔掉它影响不大。通常特征选择的准则是信息增益或信息增益比。

如果一个特征具有更好的分类能力，或者说，按照这一特征将训练数据集分成子集，使得各个子集在当前条件下有最好的分类，那么就更应该选择这个特征。

信息增益

为了便于说明，先给出熵和条件熵的定义。

熵(entropy)是表示随机变量不确定性的度量。设X是一个取有限个值的离散随机变量，其概率分布为

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

则X的熵定义为

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

显然，熵只依赖于X的分布，而与X的取值无关，从定义可以验证

$$0\leq H(p)\leq \log n$$

设有随机变量(X,Y)，其联合概率分布为

$$P(X=x_i,Y=y_j)=p_{ij},i=1,\cdots,n,j=1,\cdots,m$$

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

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

与熵和条件熵中的概率由数据估计得到时，所对应的熵与条件熵分别称为经验熵和经验条件熵。

信息增益(information gain)表示得知特征X的信息而使得类Y的信息的不确定性减少的程度。

定义(信息增益)：特征A对训练集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

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

一般地，熵$H(Y)$与条件熵$H(Y|X)$之差称为互信息(mutual information)。决策树学习中的信息增益等价于训练数据集中类与特征的互信息。

决策树学习应用信息增益准则选择特征。给定训练集D与特征A，经验熵$H(D)$表示对数据集D进行分类的不确定性，而经验条件熵$H(D|A)$表示在特征A给定的条件下对数据集D进行分类的不确定。那么它们的差，即信息增益，就表示由于特征A而使得对数据集D的分类的不确定性减少的程度。显然，信息增益大的特征具有更强的分类能力。

选择方法是：计算每个特征的信息增益，选择最大的一个。

算法：

1. 计算数据集D的经验熵$H(D)$

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

2. 计算特征A对数据集D的经验条件熵$H(D|A)$

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

3. 计算信息增益

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

信息增益比

以信息增益作为依据，存在偏向于选择取值较多的特征的问题，使用信息增益比，可以对其矫正。

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

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

其中，$H_A(D)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\log\frac{|D_i|}{|D|}$，$n$是特征$A$取值的个数。

决策树的生成

ID3算法

ID3算法的核心是在决策树各个结点上应用信息增益准则选择特征，递归构建。具体方法是：从根结点开始，按照信息增益原则选择特征，建立子结点；再对子结点递归，直到所有特征的信息增益均很小或者没有特征可以选择为止。ID3相当于用极大似然进行概率模型的选择。

算法：

1. 若D中所有实例属于同一类C_k，则T为单结点树，并将类C_k作为该结点的类标记，返回T。
2. 若$A=\varnothing$，则T为单结点树，并将D中实例数最大的类C_k作为该结点的类标记，返回T。
3. 否则，按照上述算法计算A中各特征对D的信息增益，选择最大的特征A_g。
4. 如果A_g的的信息增益小于阈值ϵ，则置T为单结点树，并将D中实例数最大的类C_k作为该结点的类标记，返回T。

5. 否则，对A_g的每一可能值，将D分割为若干非空子集D_i，将D_i中实例数最大的类作为标记，构建子结点，由结点及其子结点构成树T，返回T。
6. 对第i个子结点，以D_i为训练集，以$A-\{A_g\}$为特征集，递归地调用1-5步，得到子树T_i，返回T_i。

ID3算法只有树的生成，所以该算法生成的树容易产生过拟合。

C4.5算法

C4.5算法与ID3类似，不过是使用信息增益比来选择特征。

算法：

1. 若D中所有实例属于同一类C_k，则T为单结点树，并将类C_k作为该结点的类标记，返回T。
2. 若$A=\varnothing$，则T为单结点树，并将D中实例数最大的类C_k作为该结点的类标记，返回T。
3. 否则，按照上述算法计算A中各特征对D的信息增益比，选择最大的特征A_g。
4. 如果A_g的的信息增益比小于阈值ϵ，则置T为单结点树，并将D中实例数最大的类C_k作为该结点的类标记，返回T。
5. 否则，对A_g的每一可能值，将D分割为若干非空子集D_i，将D_i中实例数最大的类作为标记，构建子结点，由结点及其子结点构成树T，返回T。
6. 对第i个子结点，以D_i为训练集，以$A-\{A_g\}$为特征集，递归地调用1-5步，得到子树T_i，返回T_i。

决策树的剪枝

决策树生成算法递归地产生决策树，直到不能继续为止。这样产生的树往往对训练数据分类准确，但对未知数据的分类却没有那么准确，即过拟合。过拟合的原因在于学习时过多地考虑如何提高对训练数据的正确分类，从而构造出过于复杂的决策树。策略是对生成的决策树进行简化，即剪枝(pruning)。具体地，剪枝从已生成的树上裁掉一些子树或叶结点，并将其根结点或父结点作为新的叶结点。

决策树的剪枝往往通过极小化决策树整体的损失函数来实现。设树T的叶结点个数为$|T|$，t是树T的叶结点，该叶结点有N_t个样本，其中k类样本有N_{tk}个，$k=1,2,\cdots,K$。$H_t(T)$为叶结点t上的经验熵，$\alpha\geq 0$为参数，则决策树学习的损失函数可以定义为

$$C_\alpha(T)=\sum_{t=1}^{|T|}N_tH_t(T)+\alpha|T|$$

其中经验熵为

$$H_t(T)=-\sum_{k=1}^{K}\frac{N_{tk}}{N_t}\log\frac{N_{tk}}{N_t}$$

在损失函数中，将$C_\alpha(T)$的第一项记作

$$C(T)=\sum_{t=1}^{|T|}N_tH_t(T)=-\sum_{t=1}^{|T|}\sum_{k=1}^{K}N_{tk}\log\frac{N_{tk}}{N_t}$$

这时有

$$C_\alpha(T)=C(T)+\alpha|T|$$

$C(T)$表示模型对训练数据的预测误差，即模型与训练数据的拟合程度，$|T|$表示模型复杂度，参数α控制影响。

剪枝，就是当α确定时，选择损失函数最小的模型。当α值确定时，子树越大，往往与训练数据拟合越好，但是模型复杂度越高；相反，子树越小，模型的复杂度越低，但是与训练数据拟合不好。

可以看出，决策树生成只考虑了通过提高信息增益（或信息增益比）对训练数据进行更好的拟合，而决策树剪枝通过优化损失函数还考虑了减小模型复杂度。上式定义的损失函数的极小化等价于正则化的极大似然估计。所以，利用损失函数最小原则进行剪枝就是用正则化的极大似然估计进行模型选择。

树的剪枝算法：

1. 计算每个结点的经验熵$H_t(T)$
2. 递归地从树的叶结点向上回缩。

   设一组叶结点回缩到其父结点之前与之后的整体树为T_B, T_A，其对应的损失函数值为$C_\alpha(T_B), C_\alpha(T_A)$，如果

   $$C_\alpha(T_A) \leq C_\alpha(T_B)$$

   则进行剪枝，即将父结点变为新的叶结点。
3. 返回2，直到不能继续为止，得到损失函数最小的子树T_α。

CART算法

分类与回归树(classification and regression tree, CART)模型由Breiman等人在1984年提出。CART同样由特征选择、生成和剪枝组成，既可用于分类，也可以用于回归。

CART是在给定输入随机变量X条件下输出随机变量Y的条件随机分布的学习方法。CART假设决策树是二叉树，内部结点特征取值为“是”或“否”，左支为“是”，右支为“否”。

CART算法由以下两步组成：

1. 决策树生成：生成的决策树要尽量大。
2. 决策树剪枝：用验证数据集对已生成的树进行剪枝并选择最优子树。

CART生成

决策树的生成就是递归地构造二叉决策树的过程。对回归树用平方误差最小准则，最分类树用基尼指数最小化准则。

回归树的生成

假设X, Y是输入与输出变量，且Y是连续变量，给定训练数据集

$$D = \{(x^1, y^1), \cdots, (x^N, y^N)\}$$

一个回归树对应着输入空间的一个划分以及在划分的单元上的输出值。假设已将输入空间划分为M个单元$R_1, \cdots, R_M$，并且在每个单元R_m上有一个固定的输出值c_m，于是回归树模型可表示为

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

当输入空间的划分确定时，可以用平方误差$\sum\limits_{x^i \in R_m} (y^i - f(x^i))^2$来表示回归树对于训练数据的预测误差。易知，单元$R_m$上的$c_m$的最优值$\hat{c}_m$是$R_m$上的所有输入实例$x^i$对应的输出$y^i$的均值，即

$$\hat{c}_m = ave(y^i | x^i \in R_m)$$

问题是怎样对输入空间进行划分，这里采用启发式的方法，选择第j个变量x_j和它的取值s，作为切分变量和切分点，并定义两个区域：

$$R_1(j, s) = \{x | x_j \leq s\} \text{和 } R_2(j, s) = \{x | x_j > s\}$$

然后寻找最优切分变量j和最优切分点s。具体地，求解

$$\min_{j,s}\left[\min_{c_1}\sum_{x^i\in R_1(j,s)}(y^i-c_1)^2+\min_{c_2}\sum_{x^i\in R_2(j,s)}(y^i-c_2)^2\right]$$

对固定输入变量j可以找到最优切分点s

$$\hat{c}_1=ave(y^i|x^i\in R_1(j,s))和\ \hat{c}_2=ave(y^i|x^i\in R_2(j,s))$$

遍历所有输入变量，找到最优的切分变量j，构成一个对(j,s)。依次将输入空间划分为两个区域。接着，重复上述过程，直到满足条件。这样就生成了一棵回归树。这样的回归树通常被称为**最小二乘回归树**，算法如下。

算法：

1. 选择最优切分变量j与切分点s，即求解

$$\min_{j,s}\left[\min_{c_1}\sum_{x^i\in R_1(j,s)}(y^i-c_1)^2+\min_{c_2}\sum_{x^i\in R_2(j,s)}(y^i-c_2)^2\right]$$

2. 用选定的对(j,s)划分区域并决定相应的输出值：

$$R_1(j,s)=\{x|x_j\le s\},R_2(j,s)=\{x|x_j>s\}$$
$$\hat{c}_m=\frac{1}{N_m}\sum_{x^i\in R_m(j,s)}y^i,x\in R_m,m=1,2$$

3. 继续对两个子区域调用1和2，直到满足停止条件。
4. 将输入空间划分为M为区域R_m，生成决策树

$$f(x)=\sum_{m=1}^{M}\hat{c}_mI(x\in R_m)$$

分类树的生成

分类树用基尼指数选择最优特征。

定义：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

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

对于给定的样本集合D，其基尼指数为

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

如果样本集合D根据特征A是否取某一可能值a被分割为D_1,D_2两部分，即

$$D_1=\{(x,y)\in D|A(x)=a\},D_2=D-D_1$$

则在特征A的条件下，集合D的基尼指数定义为

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

基尼指数$Gini(D)$表示集合D的不确定性，基尼指数$Gini(D,A)$表示经$A=a$分割后集合D的不确定性。基尼指数越大，不确定性也就越大，这与熵相似。

算法：

0. 根据训练数据集，从根结点开始，递归地对每个结点进行一下操作。

1. 设结点的训练数据集为D，计算现在特征对该数据集的基尼指数，此时，对每一个特征A，对其可能的每个取值a，根据样本点$A=a$的测试为“是”或“否”将D分为D_1, D_2两部分，计算$A=a$的基尼指数。
2. 在所有可能的特征A和它们所有可能的切分点a中，选择基尼指数最小的特征及其对应的切分点作为最优特征与最优切分点，生成两个子结点，将训练数据集按照特征分配到两个子结点中去。
3. 对两个子结点递归地进行1和2，直到满足停止条件。
4. 生成CART决策树。

算法停止计算的条件是结点中的样本个数小于预定阈值，或样本集的基尼指数小于预定阈值(样本基本属于同一类)，或者没有更多特征。

CART剪枝

CART剪枝算法从“完全生长”的决策树的底端剪去一些子树，使决策树变小，从而提高泛化能力。由两步组成：首先从省城算法产生的决策树T_0底端开始不断剪枝，直到T_0的根结点，形成一个子树序列$\{T_0, T_1, \cdots, T_n\}$；然后通过交叉验证在独立的验证集上对子树序列测试，选择最优子树。

剪枝，形成一个子树序列

在剪枝过程中，计算子树的损失函数

$$C_\alpha(T) = C(T) + \alpha|T|$$

其中$C(T)$为对训练数据的预测误差（如基尼指数），$C_\alpha(T)$为整体损失。

对固定的α，一定存在使损失函数$C_\alpha(T)T_\alpha$最小的子树，将其表示为$T_\alpha\alpha$，且是唯一的。当α小的时候，T_α偏大；当α大的时候，T_α偏小；当$\alpha=0$的时候，整体树是最优的；当$\alpha\to\infty$时，根结点组成的单结点树是最优的。

可以用递归的方法剪枝。将α从小增大，$0<\alpha_0<\alpha_1<\cdots<\alpha_n<+\infty$，产生一系列的区间$[\alpha_i, \alpha_{i+1}),\ i=0,1,\cdots,n$；剪枝得到的子树序列对应着区间$\alpha\in[\alpha_i, \alpha_{i+1})$的最优子树序列$\{T_0, T_1, \cdots, T_n\}$，序列中的子树是嵌套的。

具体地，从整体树T_0开始剪枝，对T_0的任意内部结点t，以t为单结点数的损失函数是

$$C_\alpha(t) = C(t) + \alpha$$

以t为根结点的子树T_t的损失函数是

$$C_\alpha(T_t) = C(T_t) + \alpha|T_t|$$

当$\alpha=0$及α充分小时，有不等式

$$C_\alpha(T_t) < C_\alpha(t)$$

当α增大时，在某一α有

$$C_\alpha(T_t) = C_\alpha(t)$$

当α再增大时，不等式反向，只要$\alpha=\frac{C(t)-C(T_t)}{|T_t|-1}$，$T_t$与$t$有相同的损失函数值，而$t$的结点少，因此$t$比$T_t$更可取，对$T_t$进行剪枝。

为此，对T_0中每一内部结点t，计算

$$g(t)=\frac{C(t)-C(T_t)}{|T_t|-1}$$

它表示剪枝后整体损失函数减少的程度。在T_0中减去$g(t)$最小的T_t，将得到的子树作为T_1，同时将最小的$g(t)$设为α_1，T_1为区间$[\alpha_1,\alpha_2)$的最优子树。

如此下去，直到得到根结点，在这一过程中，不断增加α的值，产生新的区间。

在剪枝得到的子树序列中通过交叉验证选取最优子树

具体地，利用独立的验证集，测试子树序列$T_0,T_1,\cdots,T_n$中各个子树的平方误差或基尼指数。最小的决策树被认为是最优的。当最优子树T_α确定了，对应的α也确定了。

现在写出CART剪枝算法。

算法：

1. 设$k=0$，$T=T_0$.
2. 设$\alpha=+\infty$.
3. 自下而上地对各内部结点t计算$C(T_t)$，$|T_t|$以及

$$g(t)=\frac{C(t)-C(T_t)}{|T_t|-1}$$
$$\alpha=\min\{\alpha,g(t)\}$$

4. 对$g(t)=\alpha$的内部结点t进行剪枝，并对叶结点t以多数表决法决定其类，得到树T.
5. 设$k=k+1$，$\alpha_k=\alpha$，$T_k=T$.
6. 如果T_k不是由根结点及两个叶结点构成的树，则返回步骤3；否则令$T_k=T_n$.
7. 采用交叉验证法在子树序列$T_0,T_1,\cdots,T_n$中选取最优子树T_α.