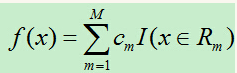
CART：Classification and regression tree，分类与回归树。（是二叉树）

CART是决策树的一种，主要由特征选择，树的生成和剪枝三部分组成。它主要用来处理分类和回归问题，下面对分别对其进行介绍。

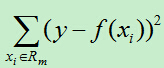
1、回归树：使用平方误差最小准则

训练集为：D={(x1,y1), (x2,y2), …, (xn,yn)}。

输出Y为连续变量，将输入划分为M个区域，分别为R1,R2,…,RM,每个区域的输出值分别为：c1,c2,…,cm则回归树模型可表示为：



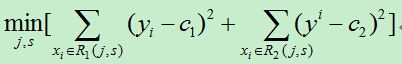
则平方误差为：



假如使用特征j的取值s来将输入空间划分为两个区域，分别为：

IMG_258

我们需要最小化损失函数，即：



　　其中c1,c2分别为R1,R2区间内的平均值。（此处与统计学习课本上的公式有所不同，在课本中里面的c1,c2都需要取最小值，但是，在确定的区间中，当c1,c2取区间平均值时其平方会达到最小，为简单起见，故而在此直接使用区间的输出均值。）

　　为了使平方误差最小，我们需要依次对每个特征的每个取值进行遍历，计算出当前每一个可能的切分点的误差，最后选择切分误差最小的点将输入空间切分为两个部分，然后递归上述步骤，直到切分结束。此方法切分的树称为最小二乘回归树。

最小二乘回归树生成算法：

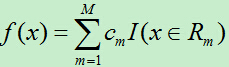
1）依次遍历每个特征j，以及该特征的每个取值s，计算每个切分点（j,s）的损失函数，选择损失函数最小的切分点。

IMG_260

2）使用上步得到的切分点将当前的输入空间划分为两个部分

3）然后将被划分后的两个部分再次计算切分点，依次类推，直到不能继续划分。

4）最后将输入空间划分为M个区域R1,R2,…,RM,生成的决策树为：

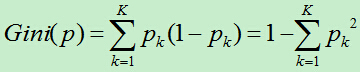


其中cm为所在区域的输出值的平均。

　　总结：此方法的复杂度较高，尤其在每次寻找切分点时，需要遍历当前所有特征的所有可能取值，假如总共有F个特征，每个特征有N个取值，生成的决策树有S个内部节点，则该算法的时间复杂度为：O(F\*N\*S)

2、分类树：使用基尼指数最小化准则

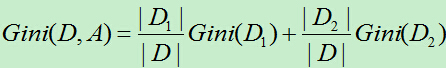
基尼指数：假如总共有K类，样本属于第k类的概率为：pk，则该概率分布的基尼指数为：



基尼指数越大，说明不确定性就越大。

对于二类分类：IMG_263

使用特征A=a，将D划分为两部分，即D1（满足A=a的样本集合），D2（不满足A=a的样本集合）。则在特征A=a的条件下D的基尼指数为：



Gini(D)：表示集合D的不确定性。

Gini(A,D)：表示经过A=a分割后的集合D的不确定性。

CART生成算法：

1）依次遍历每个特征A的可能取值a，对每一个切分点（A, a）计算其基尼指数。

2）选择基尼指数最小的切分点作为最优切分点。然后使用该切分点将当前数据集切分成两个子集。

3）对上步切出的两个子集分别递归调用1）和2）,直至满足停止条件。（算法停止的条件是样本个数小于预定阀值，或者样本集的基尼指数小于预定阀值或者没有更多特征）

4）生成CART决策树。

3、CART树剪枝

　　通过CART刚生成的决策树我们记为T0，然后从T0的底端开始剪枝，直到根节点。在剪枝的过程中，计算损失函数：

IMG_265

　　注：参数IMG_266此处为了方便编辑使用a来表示。

　　a>=0，C(T)为训练数据的预测误差，|T|为模型的复杂度。

　　对于一个固定的a，在T0中一定存在一颗树Ta使得损失函数Ca(T)最小。也就是每一个固定的a，都存在一颗相应的使得损失函数最小的树。这样不同的a会产生不同的最优树，而我们不知道在这些最优树中，到底哪颗最好，于是我们需要将a在其取值空间内划分为一系列区域，在每个区域都取一个a然后得到相应的最优树，最终选择损失函数最小的最优树。

　　现在对a取一系列的值，分别为：a0<a1<…<an<+无穷大。产生一系列的区间[ai,ai+1)。在每个区间内取一个值ai，对每个ai，我们可以得到一颗最优树Tai。于是我们得到一个最优树列表{T0,T1,…,Tn}。

　　那么对于一个固定的a，如何找到最优的子树？

　　现在假如节点t就是一棵树，一颗单节点的树，则其损失函数为：

Ca(t)=C(t)+a\*1

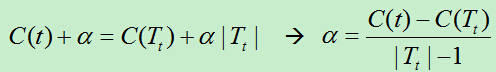
　　对于一个以节点t为根节点的树，其损失函数为：

Ca(Tt)=C(Tt)+a\*|Tt|

　　当a=0时，即没有剪枝时，Ca(t) > Ca(Tt)。因为使用决策树分类的效果肯定比将所有样本分成一个类的效果要好。即使出现过拟合。

　　 然而，随着a的增大，Ca(t)和Ca(Tt)的大小关系会出现变化（即Ca(t)- Ca(Tt)随着a单调递减。只是猜测，未经证明）。所以会出现Ca(t)= Ca(Tt)，即t和Tt有相同的损失函数，而t的节点少，故而选择t。

当Ca(t)= Ca(Tt)时，即：



　　据上分析，在T0中的每个内部节点t，计算a的值，它表示剪枝后整体损失函数减少的程度。在T0中剪去a最小的子树Tt，将得到的新的树记为T1，同时将此a记为a1。即T1为区间[a1,a2)上的最优树。

　　a与损失函数之间的关系分析：当a=0时，此时未进行任何剪枝，因为产生过拟合，所以损失函数会较大，而随着a的增大，产生的过拟合会慢慢消退，因而，损失函数会慢慢减小，当a增大到某一值时，损失函数会出现一个临界值，因而a超过此临界值继续增大的话损失函数就会因而模型越来越简单而开始增大。所以我们需要找到一个使损失函数最小的临界点a。

　　如何找到使损失函数最小的a呢？我们通过尝试的方式，依次遍历生成树的每一个内部节点，分别计算剪掉该内部节点和不剪掉该内部节点时的整体损失函数，当这两种情况的损失函数相等时，我们可以得到一个a，此a表示当前需要剪枝的最小a。这样每个内部节点都能计算出一个a。此a表示整体损失函数减少的程度。

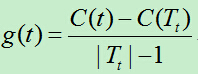
　　那么选择哪个a来对生成树进行剪枝呢？我们选择上面计算出的最小的a来进行剪枝。假如我们选择的不是最小的a进行剪枝的话，则至少存在两处可以剪枝的内部节点，这样剪枝后的损失函数必然会比只剪枝一处的损失要大（这句话表述的可能不准确），为了使得损失函数最小，因而选最小的a来进行剪枝。

　　在选出a之后，我们就需要计算该a对应的使损失函数最小的子树。即从树的根节点出发，逐层遍历每个内部节点，计算每个内部节点处是否需要剪枝。剪枝完之后的树便是我们所需要的树。

CART剪枝算法：

1）设k=0,T=T0, a=+无穷大

2）自下向上地对各内部节点进行遍历，计算C(Tt)，|Tt|及g(t)



a=min(a, g(t))；

3）自上向下访问各内部节点t，若g(t)=a，则进行剪枝，并对t以多数表决的方式决定其类。得到树T

4）若T不是由根节点单独构成的树，则重复步骤3）得到一系列的子树。

5）最后使用交叉验证的方式从子树序列中选取最优子树