研究生算法课课堂笔记

上课日期：2016 年 10 月 10 日　　第(2)节课

组长学号及姓名：高旭 1601214508

组员学号及姓名：毛琪 1601111305

组员学号及姓名：郭远 1601214509

一、 内容概要

本节课所学的内容包括以下几点：

1. 回顾偏差-方差权衡（Bias-Variance Tradeoff），包括控制决策树方差的方法，线性模型和决策树做正则化（Regularization）的方法；
2. 回顾决策树的剪枝，包括剪枝的方法和判断标准；
3. 介绍连续型属性值（Continuous Valued Attributes）的处理；
4. 介绍属性值有很多取值的情况（Attributes with Many Values）；
5. 介绍属性值分类（continuous, discrete but ordered, and categorical）；
6. 介绍有费用的属性值情况（Attribute with cost）；
7. 讨论缺失值的处理（unknown attribute values）。

二、 详细内容

1、 偏差-方差权衡（Bias-Variance Tradeoff）

Bias（偏差）: A learner’s tendency to consistently learn the same wrong thing，即度量了某种学习算法的平均估计结果所能逼近学习目标的程度。

Variance（方差）: The tendency to learn random things irrespective of the real signal，即度量了在面对同样规模的不同训练集时，学习算法的估计结果发生了变动的程度。

Bias-Variance Tradeoff 可以用图 1 来解释。图中靶心代表某个能完美预测的模型（预测结果），离靶心越远，准确率越低。靶上的点代表在数据集上学习到的某个模型，High Bias 表示离靶心偏差较大，Low Bias 表示离靶心偏差较小，High Variance 表示多次学习过程的预测结果较分散，Low Variance 表示多次学习过程的预测结果较集中。High Bias 易发生欠拟合，High Variance 易发生过拟合，

因此需要在二者之间做权衡。

决策树属于 Low Bias, High Variance 的。但是具体到某一棵决策树，其 bias 和 variance 的 tradeoff 是不一样的。

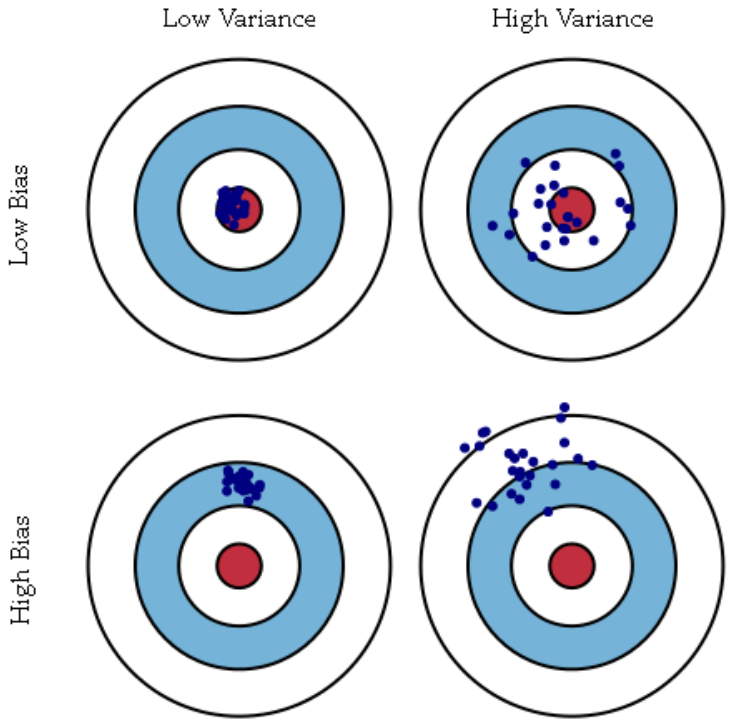

图 1　Bias-Variance Tradeoff 示意图

控制决策树 variance 的方法

(1) 控制树的高度；

(2) 控制叶子结点的个数；

(3) 样本顺着决策树向下分类，设置节点允许的最少样本数作为阈值。（例如每个节点至少需要五个样本，如果不到五个样本就不能继续向下分了）

(4) 考虑将子节点允许的最少样本数设为阈值。（不考虑当前有没有五个样本，而是看分类后的每个子节点有没有五个样本，如果某个子节点不到五个节点，就不能继续向下分了）

凡是控制 Variance 的方法都属于 **regularization(正则化)，目的是控制模型的复杂度**。线性回归模型一般来说是 bias 比较强的，但是如果特征属性非常多，也很容易出现过拟合。比如研究生招生的例子，训练样本是 1000 个待招生的学

生，如果每个学生有 2000 个属性，那么即使用线性模型也很容易出现过拟合。

线性模型和决策树做 regularization 的方法

线性模型做 regularization 的主要方法是控制权重。对于线性回归模型：$y = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$，一种 regularization 的方法是控制$|w|$的大小，拟合的时候不让$|w|$过大。另一种方法是控制训练的时间。一般出现过拟合，都是因为训练时间过长。这一点在深度学习中也很适用。

决策树做 regularization 的主要方法是控制树的高度，即剪枝。一个极端例子，对于决策树桩（Decision stump，只有一个节点的树），此时其 bias 很大，variance 几乎为 0，意味着训练数据集只能分类一次就需要得出结果。在树桩的位置，样本量大，即使存在一些噪声（noise），也不容易受到干扰。但是随着树的生长，每个节点的样本数在减少，到了树枝，可能只剩下一两个样本，这时就很容易受到噪声干扰。

Bias 比较大的算法之优缺点

Bias 比较大的算法，优点是比较稳重，缺点是无法捕捉细节，刻画不出较为复杂的分类面。例如在研究生招生中，如果学生之间的差异很大，要么一看肯定就能录取（机试做了 9 道题），要么一看肯定不能录取，那么比较矮的决策树就可以刻画该模型。但是如果两个学生彼此相差不大（GPA 相差无几，都有各种竞赛加分，都喜欢扔铅球等等），那么这种 bias 大的算法就不能区分了。

决策树是非参数的

决策树（decision tree）是非参数的（non-parametric）。注意：决策树可以有超参数，即控制模型本身复杂度的参数（例如树的高度），但是没有像线性回归那样与属性值相乘的参数。**下面具体解释下参数分类。**

参数分为两种，一种是像线性回归那样的普通参数，

$$y = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

其中的参数$(w_0, w_1, w_2, \ldots, w_n)$是跟数据的属性直接相乘。

另一种是像决策树那样，是模型本身的参数。例如是控制模型的复杂度（树的高度），或者是控制算法的配置（熵/Gini 指数/错误率）。

2、 决策树的剪枝

Q1: 决策树剪枝的方法是什么?可以只剪子类的一个吗？

在处理决策树的剪枝的时候，如果被分成几个子类，那么这几个子类只能一起剪掉，相当于把这几个子类合并到父类，不可能只剪掉其中一个子类。例如图 2 所示，我们剪掉左图的“色泽”这一个属性变成右图的决策树。

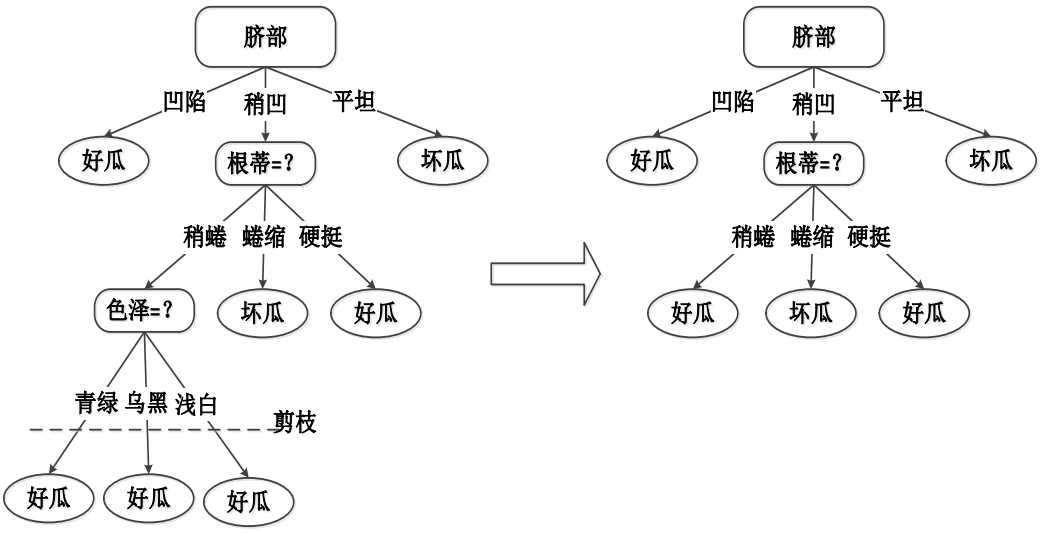

图 2　一个决策树剪枝的实例

Q2-1: 决策树剪枝的标准是什么？

我们将数据集分为训练集（Training Set）、验证集（Validation Set），测试集（Test Set），一般来说数据集的比例可以大致分为训练集 60%、验证集 20%和测试集 20%。示意图见图 3。

利用训练集训练结束生成决策树之后，在处理剪枝问题的时候，需要在验证集（Validation Set）上进行判断：在验证集（Validation Set）上，若将该枝剪掉以后验证集上的分类准确率上升，则判断该枝可以剪。

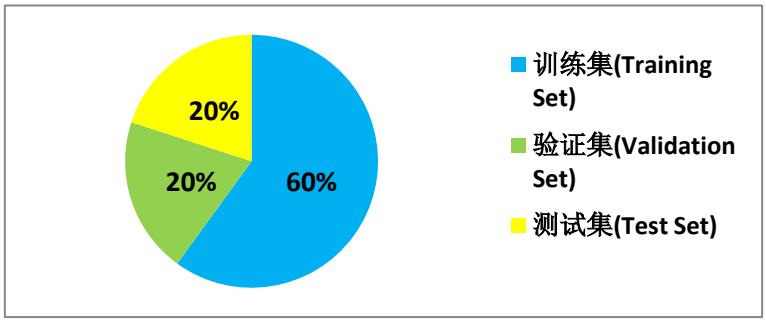

图 3　数据集拆分比例图

Q2-2: 如果按照这种情况剪枝后，在训练集上的准确率如何变化？

一定下降!

有一种说法是：在保持训练集准确性不变的情况下，如果在验证集（Validation Set）的分类会提高则剪枝，这种说法是不对的。因为我们之前的决策树的生成是通过训练集（Training Set）训练出来的，也就是说决策树中，选择某个属性进行划分（长出这个枝），是因为利用这个属性划分之后，准确率是上升的。所以剪掉该枝（去掉这个属性的划分），准确率一定是下降的。这也是机器学习上的一个很常见的方法：牺牲在训练集上的准确性，来提升算法的泛化能力。

在之后布置的大作业中，也需要注意如果将训练集上的错误率（error rate）降到零，一般来说是不好的，因为这样会把训练集中的噪音也拟合进去了。因此，训练集上的错误率（error rate）不要降得太低。

3、 连续型属性值的处理（Continuous Valued Attibutes）(PPT 71)

我们前面接触到的属性都是离散型的：比如 outlook 可能的取值有 Sunny、Overcast 和 Rain。这些属性的值是有限种可能性，在分类的时候比较容易处理。但是对于那些连续型的属性，比如说像 Temperature，它的取值是连续的，有无限种可能性，该如何利用这种属性进行划分呢？

我们采用的基本思想是将连续型属性转换成一个 binary 的离散型变量。我们计算出一个阈值（相当于是切一刀），小于这个阈值的被分在树的一边，大于这个阈值的被分在树的另外一边。比如 PPT 上的 Temperature，这就可以转换成一个新的属性，这个属性就是 Temperature 是不是大于 72.3 度。

Q: 这个阈值怎么取呢？

最简单的策略:

连续变量虽然有无限多个取值，但是训练集上的样本数是有限的。对于一个给定的训练集，这里的样本值只有有限多个，所以一个最简单的方法是在每一种可能的取值之间切一刀，比如针对第 71 页上讲义的 Temperature 来说：

Temperature	**40**	**48**	**60**	**72**	**80**	**90**
Play Tennis	**No**	**No**	**Yes**	**Yes**	**Yes**	**No**

（1） 小于 40 是一种情况，大于 40 是一种情况（40 这里做一个切分点）；

（2） 小于 44（40 和 48 之间）算一类，大于 44 算另一类（44 这里做切分点）；

（3） 以此类推。

这种方法实际上是一种采用二分法（bi-partition）对连续属性进行处理，这也是 C4.5 决策树算法中采用的机制。具体来说：

给定样本集 D 和连续属性 a，假定 a 在 D 上出现了 n 个不同的取值，将这些值从小到大进行排序，记为 $\{a^1, a^2, ..., a^n\}$.基于划分点可将 D 分为子集 D_t^- 和 D_t^+，其中 D_t^- 包含那些在属性 a 上取值不大于 t 的样本，而 D_t^+ 则包含那些在属性 a 上取值大于 t 的样本。对于相邻属性取值 a^i 与 a^{i+1} 来说，t 在区间 $[a^i, a^{i+1})$ 中任意值所产生的划分结果相同。因此，对连续属性 a，我们可选择包含 n-1 个元素的候选划分点集合：

$$T_a = \{\frac{a^i + a^{i+1}}{2} | 1 \le \mathrm{i} \le \mathrm{n}-1\}$$

我们把 $[a^i, a^{i+1})$ 的中位点 $\frac{a^i + a^{i+1}}{2}$ 作为候选划分点。然后，我们就可像离散属性值一样来观察这些划分点，选取最优的划分点进行样本集合的划分。

我们选出来了这些候选点，怎么选最优的划分点？仍然可以采取之前的评价标准，选择信息增益最多的候选点作为最优解。

优化的思路:

从直观上来看，如果切一刀，应该选择在预测值发生变化的时候切，这样做可能的效果更好。比如说像图中的 Temperature 很低的几个值比如 40、48，这些

值对应的分类都是 No，那对于这些样本，在前面切（比如 44 这里切）的候选点就没有什么意义。因此，优化的思路是我们认为获得最佳属性的一定是在 Y 值分类值发生变化的地方，并且选择这些 Y 值发生变化的属性值之间的值作为划分点。这样与之前相比，候选点数目大大降低，然后在这些切分点中根据之前的评价准则（比如信息增益准则）选择最优的切分点。

4、 属性值有很多取值（Attributes with Many Values）（PPT 72）

我们选择某一个属性时，考察的标准是希望区分度越大越好。但是以这种考察标准会存在一个问题，举例来说：

当我们预测研究生是否会被录取这个问题时，区分度最大的属性是学生的身份证号，每一个人的身份证号都是唯一的。根据这个属性进行划分会生成很多子节点，每一个子节点上只有一个样本，这样的划分肯定是非常纯的。可是很显然这种划分是没有意义的，泛化能力基本上为零，如果新来一个学生，身份证号也是唯一的，与之前的都不一样，也无法作相应的预测。这样的学习（训练）是没有意义的。

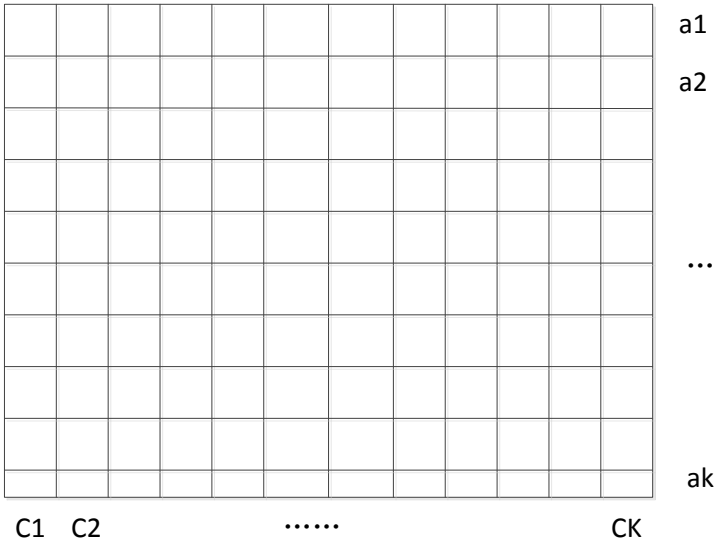

图 4　多属性值划分数据集示意图

如图 4 所示，竖线的划分表示类型（目标值）的划分，分为$C_1 \dots C_k$，横线的划分表示属性 a 的所有取值的划分。我们可以看到每一个小格子表示的就是当前属性a_k取值下属于C_k这个类型的样本集，从直观上来看如果所取的属性的取值越多，划分得越细，样本的数目也就越少，所以样本倾向的纯度也会越高。那如果

直接利用信息增益的准则来选择最优的属性，我们更倾向于选择这些取值情况有很多的属性。对于这类属性可能会存在类似于身份证号这种区分度很小的可能性。因此考虑采用类似于正则化的思想，加一个惩罚项来对信息增益进行调整，如下式所示，称这个为信息增益比（Information Gain Ratio）

$$\text{Gain}Ratio(S,\text{A}) = \frac{Gain(S,A)}{SplitInformation(S,A)}$$

$$SplitInformation(S,A) = -\sum_{i=1}^{c}\frac{|S_i|}{|S|}\log_2\frac{|S_i|}{|S|}$$

其中S_i是数据集S中属性A取值为v_i的划分。

在与之前的信息增益（Information Gain）相比，需要除以一个惩罚项：Split Information，目的就是为了惩罚那些取值太多的属性。

分裂信息（Split Information）

分裂信息并不考虑类别的划分，而只是考虑属性值的划分，这个项的结构类似于熵，按照这个属性去进行划分，可以看做是分裂信息的熵。

如果有 S 个样本按照属性 A 去划分，划分有 C 种可能性，属性 A 的取值有 C 种，每一种的样本数是 S_i，得到的概率去求熵。从上述式子可以看出，属性划分的可能性越多，分裂信息的熵越大，惩罚项越大，得到最终的信息增益越小。

实际情况的思路

实际情况下如果直接使用这种方法可能会出现另外一种极端：某种属性没有多少区分度，但是取值比较少，比如只有两种取值。算出来的分裂信息的熵（Split Information）比较小，信息增益比（Information Gain Ratio）比较大；

另外一种属性，它是有区分度的（与身份证号属性不同），但它的取值同样比较多，比如有 7 种取值。分裂信息的熵（Split Information）比较大，信息增益比（Information Gain Ratio）可能就会变得很小。这样属性选择的时候，可能倾向于选择取值比较少的属性。

所以实际情况下一般是采取这样一种策略：

（1） 先按照信息增益（Information Gain），选出排在前面的比如 50%的属性值。

（2） 然后在 50%的这些属性里面，再按照信息增益比（Information Gain Ratio）选出最优的属性。

按照这种做法，对于那些本来属性取值很少又没有多少区分度的属性，在第一步就可以排除掉，剩下到第二步的属性都是有区分度的，再根据属性取值多少计算信息增益比（Information Gain Ratio）进行比较，从而可以避免两种极端情况。

决策树是二叉树

实际上用的更多的情况是只允许切一刀，无论属性有多少取值，只允许变成二叉树。比如 CART（classification and regression tree， 可以用作分类和回归，其他算法如 ID3，C4.5 主要是用来分类），每次只能切一刀。

这种情况下做法如下：如属性天气的取值有 3 个：sunny、overcast、rainy。第一次划分将属性划分为 sunny、not sunny，第二次将 not sunny 划分为 overcast、rainy。如图 5：

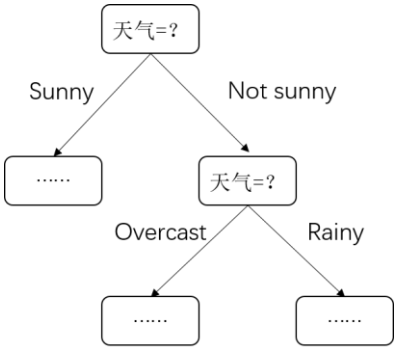

图 5 示意图

5、 属性值的分类

变量的属性可以分为三类：

（1） Continuous（也叫 numerical）： 连续型变量。如：GPA、Temperature；

（2） Discrete but ordered： 离散型有顺序的变量，比较大小有意义。如：机考时答对的题数、降水概率（晴天、阴天、雨天）；

（3） Categorical： 类型变量。比较大小无意义。如：专业、身份证号、学号等。

Categorical 变量处理

如果决策树分叉个数不限制：多少个值多少分叉。

如果决策树必须是二叉树：每次切一刀，没有明确的意义（另外两种 continuous 变量和 discrete but ordered 变量，每次切一刀有实际意义）。正规做法为 one-hot encoding。

One-Hot encoding

One-hot encoding 将属性的 N 个取值，转换成一个 N 维长向量，在任意时刻只有一位为 1，其余位为 0。也就是将一个属性变成了 N 个属性。如某个样本集中，1000 个样本有 1000 个不同的身份证号，通过 one-hot encoding，身份证号属性变成一千个属性。每个属性代表一个身份证号，值为 1 代表他的身份证号是这个号，值为 0 代表他的身份证号不是这个号，一个样本中，只能有 1 位是 1，其余位为 0。

实际情况

实际应用也可以就用二叉树，每次切一刀，切得足够细，足够深，最后会有意义。比如专业，初始划分时没有实际意义，但划分到最后，每个叶节点就是各个专业名称，计算机系、数学系…，如图 6 所示。

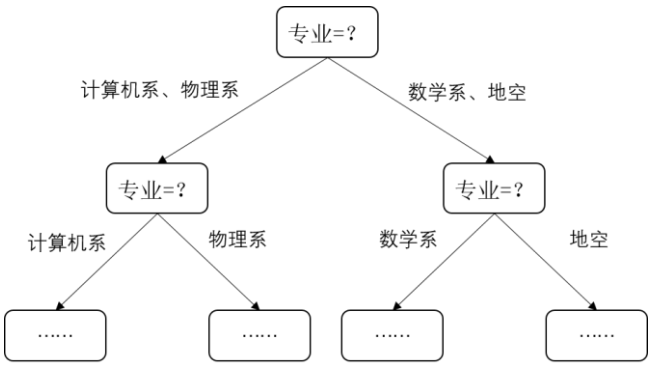

图 6　专业划分示意图

6、 有费用的属性值（Attribute with cost）

选择划分属性时，最有区分度的属性不一定是最好的。比如看病做检查，有的检查代价高、对身体有伤害，不严重的病不需要做这种检查。但是这些检查可能区分度最高，很容易被选择为划分属性，所以不能只看区分度。一些检查区分度不低，而且价格不高、身体伤害不大，就应该作为划分属性。所以引入 cost，

每个属性有一个 cost。在选择属性时，尽可能选择区分度高、cost 低的属性。引入 cost 后，每次属性选择既考虑区分度，也考虑代价。

几个考虑到 cost 后的增益计算方法：

（1） Tan and Schlimmer：$\frac{Gain^2(S,A)}{Cost(A)}$

（2） Nunez：$\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$

几个公式没有严格的理论依据，但有相同的基本原则：用 cost 做分母，抵消一部分 gain。

7、 缺失值的处理（重要）

样本的每个属性的值不一定是全的，有缺失值是常态（比如，研究生录取，排名有的没有，四六级成绩不一定都有。看病时，病人不可能把所有检查做一遍。）。

解决方法如下：

（1） 选择大多数，看其他样本在这个属性大部分取哪些值，就取相同的值。

（2） 与（1）类似，但只看同类样本中的值，选择同类样本中的大多数。

（3） 根据该节点该属性值的概率分布赋值。