第五讲 决策树模型

2025年3月18日

概要

依规则分类

- 日常生活中常用的分类方法之一就是使用规则对所关心的对象进行分类,比如
  - “三色为俏，五色为宝”的玛瑙分类规则.
  - 中药材按照“七两为参八两为宝”对人参依据重量进行等级区分.
- 分类规则的前件或者头部通常是由若干关于对象的特征（属性）检测的合取所构成的条件, 例如
  - 人参的重量是否超过七两，且人参须子品相是否完好等,
- 规则的后件或尾部则是对满足规则前件的对象分配的类别。
- 通常要求用于分类的规则集是互斥并且完备的.

依规则分类

- 通常某些比较重要的属性测试条件被若干条规则所共有，比如
  - 如果人参重量超过八两，且人参须子品相完好，则为上品（规则R_1）
  - 如果人参重量超过八两，但人参须子品相略有损耗，则为中品(规则R_2)

  这两条规则中共享条件“重要是否超过八两”.
- 一般来说，某些条件被越多的规则所共有，则对应的属性对于分类来说越重要.
- 我们可以采用有向树形结构来对规则集合进行一个约简表示.

依规则分类

- 用内部结点和叶子结点表示属性和类别.
- 内部结点到其孩子结点的边信息表示对属性的检测信息.
- 从根结点到每个叶子结点的路径表示一条规则:
  - 从根到叶结点的路径所经过的属性检测构成规则的前件.
  - 叶子结点对应的类别构成规则的结论.
  - 规则之间共享的条件通过共享路径来表示.

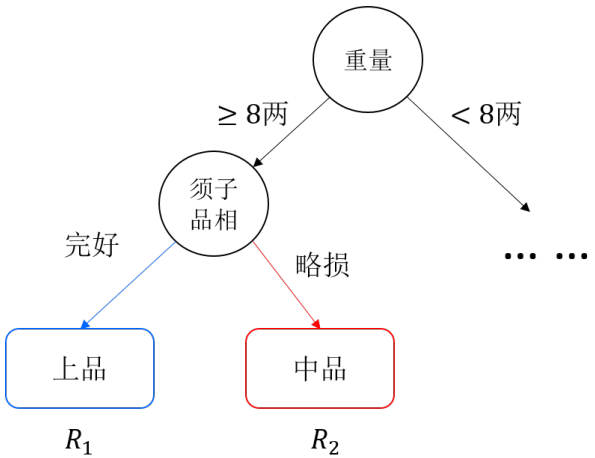

Figure: 规则的树形表示

概要

特征选择

- 分类规则是基于特征测试来构建.
- 选择特征是分类规则学习的核心之一.
- 应该优先选择对分类最具效果的特征!

如何考察某维给定的特征的分类能力呢?

- 直观上可以按照这个特征的取值对训练数据集进行一个划分,
- 考察划分以后的数据子集在当前条件的分类情况.

- 暂时不妨假定数据的每维特征都是离散型，并且可能的取值个数有限.

经验熵

- 给定训练数据集$D=\{(x_i,y_i)\}_{i=1}^N$，其中
  - $x_i=\left(x_i^{(1)},x_i^{(2)},\cdots,x_i^{(n)}\right)^T\in\mathcal{X}$为输入示例，
  - n为特征个数或者维数，$x_i^{(j)}$为x_i的第j维特征值，
  - $y_i\in\mathcal{Y}=\{c_1,c_2,\ldots,c_K\}$为类标记，

  $i=1,2,\ldots,N;j=1,2,\ldots,n.$
- 将D划分成$D_1,D_2,\ldots,D_K$，其中D_k表示D中属于类c_k的样例的集合，$k=1,2,\ldots,K.$
- 定义训练数据集D的经验熵$H(D)$如下：

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

经验熵刻画了数据集D的纯度，如果D中的数据都属于同一类别，此时经验熵达到最小。

条件经验熵

给定某维特征A, 其取值集合为$\{a_1, a_2, \cdots, a_m\}$.

- 根据特征A的取值将D划分为m个子集合$D_1^A, D_2^A, \ldots, D_m^A$, 其中$D_i^A$ 中的示例的特征A的取值为a_i, $i = 1, 2, \ldots, m$.
- 进一步考察每个D_i^A中的分类情况, 以D_{ik}^A表示D_i^A中属于类c_k的样本的集合, 显然 $\sum_{k=1}^{K} D_{ik}^A = D_i^A$ 且 $\sum_{i=1}^{m} D_{ik}^A = D_k$.
- D_i^A的经验熵: $H(D_i^A) = -\sum_{k=1}^{K} \frac{|D_{ik}^A|}{|D_i^A|} \log_2 \frac{|D_{ik}^A|}{|D_i^A|}$.
- 用基于特征A划分的子集的平均经验熵 (即给定A 的条件下D的条件经验熵)

$$H(D|A) = \sum_{i=1}^{m} \frac{|D_i^A|}{|D|} H(D_i^A)$$

来作为给定特征A的条件下数据集D的纯度度量.

数据集划分与经验熵

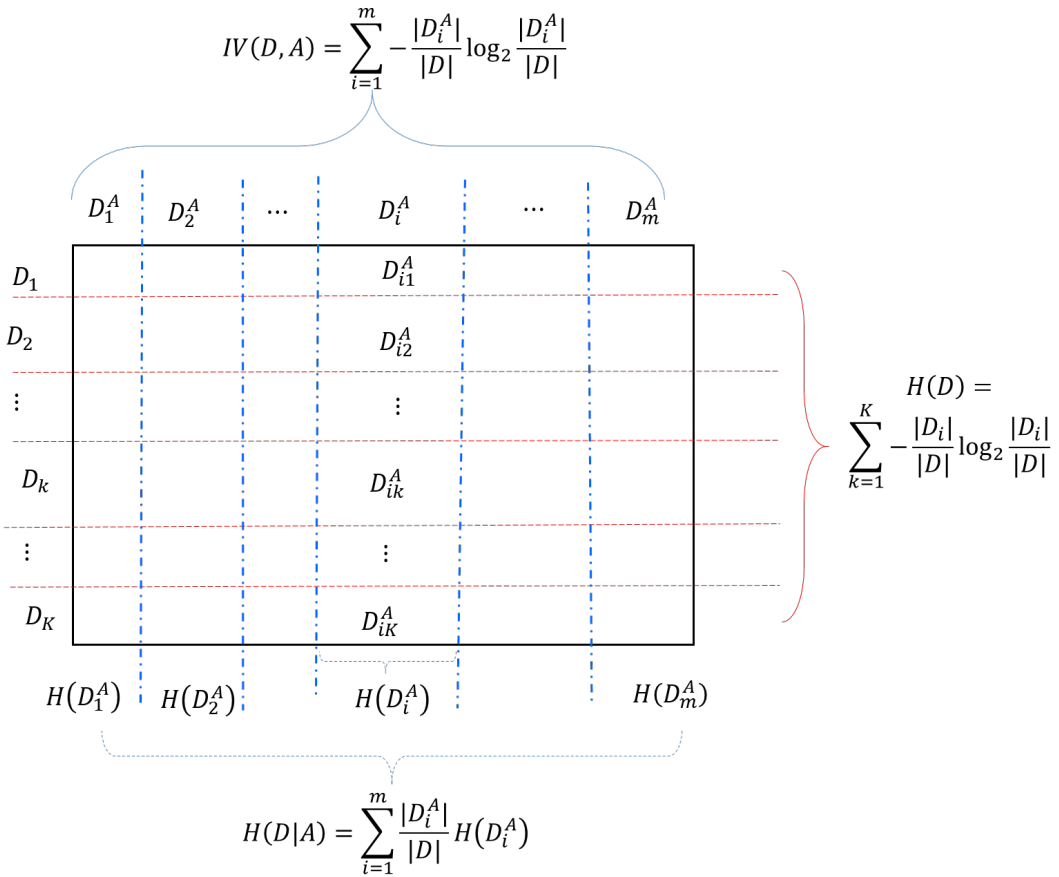

Figure: 特征A对D的划分和各划分所相应的经验熵

信息增益

信息增益(Information Gain)

属性A对样本集D进行划分所获得的信息增益 $G(D,A)$定义

$$G(D,A) = H(D) - H(D|A).$$

- 信息增益$G(D,A)$刻画的是使用特征A 来进行划分使得数据集D在纯度方面的提升幅度，在一定程度上刻画了特征A的分类能力.
- 可以使用信息增益来选择相应的特征.
- 另一方面，信息增益度量更偏向于可能取值个数比较多的特征.
  - 用分裂信息进行校正?

信息增益率

- 特征A的分裂信息IV(D,A)(也称固有值)定义如下：

$$IV(D,A) = -\sum_{i=1}^{m} \frac{|D_i^A|}{|D|} \log_2 \frac{|D_i^A|}{|D|}.$$

信息增益率(Gain Ratio)

信息增益率(Gain Ratio)$G_R(D,A)$定义为

$$G_R(D,A) = \frac{G(D,A)}{IV(D,A)}.$$

信息增益率的计算

计算特征A的信息增益(率)过程中需要对训练样本集进行的各种划分和各划分所相应的经验熵，其中

- $H(D)$是按照标记类别对D所做的划分$D_1, D_2, \cdots, D_K$ 所对应的熵;
- 分裂信息$IV(D, A)$是按照特征A的取值对D所做的划分$D_1^A, D_2^A, \ldots, D_m^A$所对应的熵;
- $H(D_i^A)$是按照标记类别对D_i^A所做的划分 $D_{i1}^A, D_{i2}^A, \cdots, D_{iK}^A$所对应的熵;
- $H(D|A)$是按照划分 $D_1^A, D_2^A, \ldots, D_m^A$ 对$H(D_1^A), H(D_2^A), \cdots, H(D_m^A)$的平均。

数据集划分与经验熵

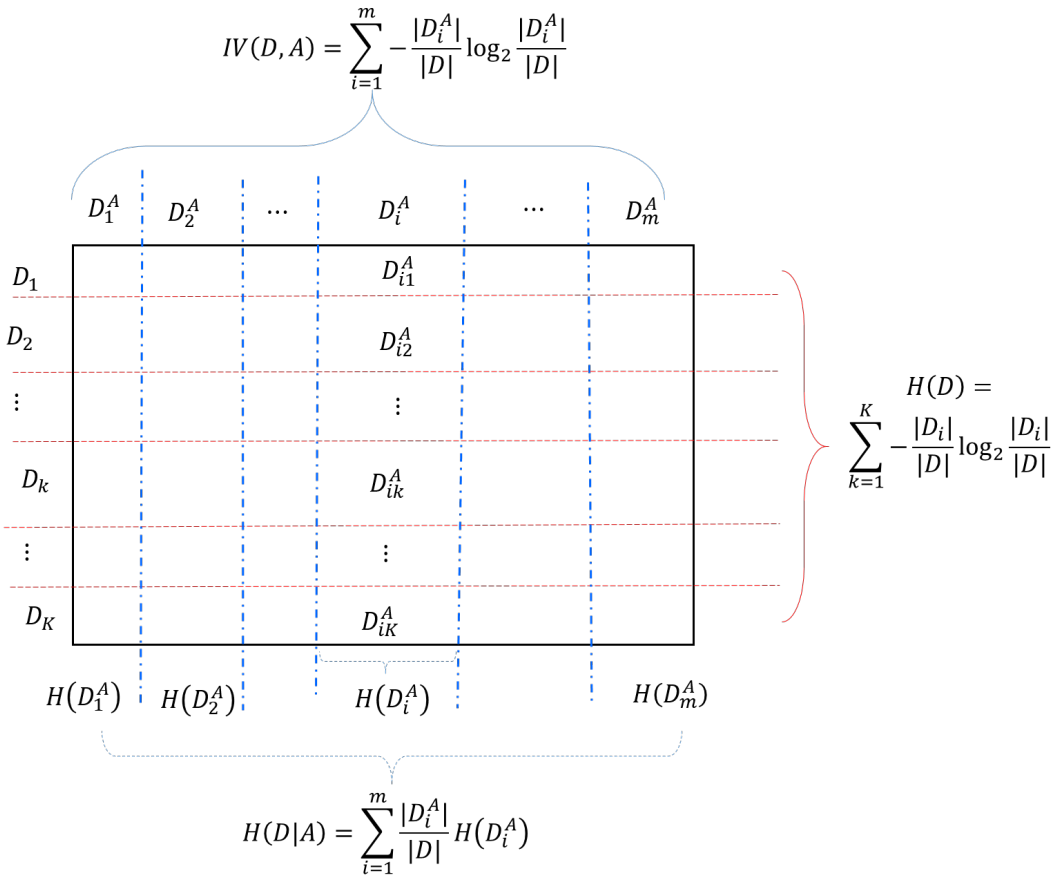

Figure: 特征A对D的划分和各划分所相应的经验熵

基尼(Gini)指数

数据集D的基尼指数$\mathrm{Gini}(D)$

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

- $\left(\frac{|D_k|}{|D|} \right)^2$ 反映了从D中随机抽取两个样本都属于c_k的概率.
- $\mathrm{Gini}(D)$可以看作是从数据集中随机抽取两个样本，其类别标记不一致的概率.

基尼(Gini)指数

- 根据特征A的取值将D划分为m个子集合$D_1^A, D_2^A, \ldots, D_m^A$.
- 特征A的基尼指数$\mathrm{Gini}(D, A)$定义为

$$\mathrm{Gini}(D, A) = \sum_{i=1}^{m} \frac{|D_i^A|}{|D|} \mathrm{Gini}(D_i^A).$$

- $\mathrm{Gini}(D, A)$越小，特征A的分类能力越强

基尼(Gini)指数

- 按照特征A是否取值为a_i对数据集D进行二分:

$$D = D_i^A \cup [D - D_i^A].$$

- $A = a_i$的基尼指数$\mathrm{Gini_d}(D, A = a_i)$定义为

$$\mathrm{Gini_d}(D, A = a_i) = \frac{|D_i^A|}{|D|}\mathrm{Gini}(D_i^A) + \frac{|D - D_i^A|}{|D|}\mathrm{Gini}(D - D_i^A),$$

$i = 1, 2, \ldots, m.$

- $\mathrm{Gini_d}(D, A = a_i)$越小，特征$A = a_i$的分类能力越强.

概要

决策树

- 决策树一般包含一个根结点、若干内部结点和叶子结点.
- 根结点包含所有样本集.
- 非叶结点和其子结点构成的子树对应于一个特征测试:
  - 根据测试结果将该结点包含的样本集合划分到其孩子结点.
- 叶结点对应于分类结果.
- 从根结点到每个叶结点的路径对应一条分类规则.
  - 每个非叶子结点所对应的特征测试结果的合取构成了规则的条件.
  - 叶子结点则对应于该规则的分类结果.

- 大多数决策树的生成过程都是从根结点开始.
- 挑选相对于当前数据集分类能力最强的特征.
- 基于最优特征对当前数据集进行划分，构建子结点.
- 再对子结点递归调用此划分方法，构建决策树.
- 直到
  - **无特征可选** 或者
  - 当前数据集已经**完全纯化**

  为止.

决策树学习算法基本框架

输入： 训练数据集$D=\{(x_i,y_i)\}_{i=1}^N$，特征集合$\mathcal{A}$，最优特征选择函数$F(\cdot,\cdot)$

输出： 决策树T

(1) 若D中样本属于同一类别c_k，则生成一个类标记为c_k的单结点树T，返回T;

(2) 若$\mathcal{A}=\emptyset$, 且D非空，则生成一个单结点树T，并以D中样本数最多的类别作为该结点的类标记，返回T;

(3) 计算$A^*=F(D,\mathcal{A})$.

(4) 对A^*的每个可能取值a_i，构造一个对应于D_i的子结点;

决策树学习算法基本框架

(5) 若D_i为空，以D中样本数最多的类别作为该子结点的类标记，并将该子结点标记为叶子结点；

(6) 否则,将D_i中样本数最多的类别作为该子结点的类标记;

(7) 由该结点及其子结点构成树T， 返回T;

(8) 对每个D_i对应的非叶子结点，以D_i为训练集，以$A-\{A^*\}$为特征集，递归地调用步(1)-(6)，得到子树T_i，返回T_i.

决策树学习算法

- 如果以信息增益最大作为选择最优特征的准则，即

$$A^* = \underset{A \in \mathcal{A}}{\operatorname{argmax}}\, G(D, A),$$

则上述算法对应于**ID3决策树学习算法**:

  - Quinlan, J.R., Induction of decision trees, *Machine Learning*, 1(1), 81-106, 1986

- 如果以信息增益率最大作为最优特征的选择准则，即

$$A^* = \underset{A \in \mathcal{A}}{\operatorname{argmax}}\, G_R(D, A),$$

则上述算法对应于**C4.5决策树学习算法**.

多路划分与二路划分

- 多路划分:按照最优特征的可能取值个数来确定数据子集的个数.
- 二路划分: 采用以特征的可能取值为切分点的二分法划分当前数据集:
  - 选择基尼指数最小的特征及其切分点$A^* = a^*$作为当前最优特征和最优切分点，即

$$(A^*, a^*) = \underset{A \in \mathcal{A}, a \in V(A)}{\operatorname{argmin}} \operatorname{Gini}_{\mathrm{d}}(D, A = a),$$

    这里$V(A)$ 是A的所有可能取值的集合.
  - 对应于 **CART（Classification and Regression Tree）算法**.
    - Leo Breiman: CART算法[1984]、Bagging模型[1996]和随机森林模型[2001].

多路划分与二路划分

- CART算法采用二分法:
  - 对应的决策树是二叉树.
  - 一般来说比多路划分算法产生更深的决策树.
  - 这会使得学习算法的时间复杂性提高.
- 相关研究表明采用不同的最优特征选择准则
  - 对决策树的规模有影响,
  - 但对其泛化能力影响不大

决策树剪枝

决策树剪枝: 降低决策树模型过拟合风险

- 预剪枝策略:在决策树的生成过程中，基于验证集对当前结点可能进行的划分进行评估.
  - 如果划分并不能带来决策树泛化性能提升，则直接将当前结点标记为叶结点.
  - 提前禁止一些分支展开也可能带来欠拟合的风险.
- 后剪枝策略: 在决策树生成以后简化决策树模型，降低过拟合风险.
  - 从树上剪掉一些子树或叶结点，并将其根结点或者父结点作为新的叶结点.
  - 对决策树剪枝前后的泛化性能进行比较决定是否剪枝.
  - 可以通过验证集来进行模型的泛化性能评估.

决策树剪枝

后剪枝: 也可以采用正则化方法

- 决策树T的损失或代价函数:

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

这里

- $C(T)$是衡量T对D的拟合程度的函数，比如可以定义为

$$C(T) = \sum_{t=1}^{|T|} N_t H_t(T),$$

其中N_t和$H_t(T)$ 分别表示叶子结点t 对应的数据集的大小和经验熵.

- T的叶结点个数$|T|$作为T模型复杂度.
- $\alpha \geq 0$是用以权衡T的拟合程度和模型复杂度的系数.

决策树剪枝

CART算法剪枝处理比较特别:

- 从CART 算法生成的完整决策树T_0开始
- 剪枝算法产生一个递增的权衡系数序列

$$0 = \alpha_0 < \alpha_1 < \cdots < \alpha_n$$

和一个嵌套的子树序列$\{T_0, T_1, \cdots, T_n\}$ 使得

  - T_i为$\alpha \in [\alpha_i, \alpha_{i+1})$所对应的最优子树，且
  - T_n 是由根结点单独构成的树.

- 通过交叉验证法在子树序列$\{T_0, T_1, \cdots, T_n\}$中挑选出最优子树.

基于连续特征的划分:二分法

给定训练数据集D和连续特征A

- 将特征A在数据集D中出现的取值从小到大的排列为

$$a_1, a_2, \cdots, a_l.$$

- 构建可能的切分点

$$s_i = \frac{a_i + a_{i+1}}{2}, \ 1 \leq i \leq l-1,$$

- 基于每个切分点s_i可以将D分为两个子集$D_{s_i}^-$和$D_{s_i}^+$，其中
  - $D_{s_i}^-$ 包含特征A的取值不超过a_i 的样本,
  - $D_{s_i}^+$ 包含特征A的取值不小于a_{i+1}的样本.
- 通过比较不同切分点对应的数据集划分的纯度度量来挑选出该特征的最优切分点.

叶结点类标记规则与经验风险最小化

- 以每个单元中样本数最多的类别作为该结点的类标记.
- 采用0-1损失函数， 不妨设给叶结点t设置的类标记为c_k，则在t对应的数据集D_t 上的经验风险为

$$\frac{1}{N_t}\sum_{x_i\in D_t} I(y_i \neq c_k) = 1 - \frac{1}{N_t}\sum_{x_i\in D_t} I(y_i = c_k).$$

则

$$\min_{c_k\in\mathcal{Y}} \frac{1}{N_t}\sum_{x_i\in D_t} I(y_i \neq c_k)$$

等价于

$$\max_{c_k\in\mathcal{Y}} \frac{1}{N_t}\sum_{x_i\in D_t} I(y_i = c_k).$$

多变量决策树（Multivariate Decision Tree）

- 决策树的构造过程划分的单元都是“矩形”的:
  - 分类边界是由若干个与特征坐标轴平行的分段边界组成.
  - 分段边界与特征切分点之间的对应使得分类规则比较直观、易解释.
  - 逼近比较复杂的分类边界需要大量的特征测试来构造比较复杂的决策树.
- 多变量决策树（Multivariate Decision Tree）
  - 允使用若干特征的**线性组合**作为切分当前数据集的“最优特征”.
  - 相当于对每个非叶结点学习一个合适的线性分类器.
  - 可以用斜的划分边界来逼近复杂的真实分类边界.

概要

最小二乘回归树

CART算法用于回归问题: 使用平方误差最小准则来选择属性和相应的最优切分点.

- 给定训练数据集$D = \{(x_i, y_i)\}_{i=1}^{N}$，其中$x_i \in \mathrm{R}^n$, $y_i \in \mathrm{R}$, $i = 1, 2, \cdots, N$.
- 回归树模型对应于输出空间的划分以及每个划分单元上的输出值:
  - 设回归树将输入空间划分成M 个单元$R_1, R_2, \cdots, R_M$，且 单元R_m 上的输出值为c_m，则回归树模型可以表示成

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

  - 如果采用平方误差最小准则，则R_m的最优输出值$\hat{c}_m$为R_m中所有实例x_i对应的y_i的平均值，即

$$\hat{c}_m = \mathrm{ave}(y_i | x_i \in R_m).$$

最小二乘回归树

- 从输入空间开始，按照平方误差最小准则递归地挑选每个区域的最优切分特征变量和最优切分点.
- 对输入空间，若选用第j维特征变量作为切分变量， s作为切分点:
  - $R_1(j,s)=\{x|x^{(j)}\leq s\}$ 和 $R_2(j,s)=\{x|x^{(j)}>s\}$,
  - 通过求解

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right] \tag{1}$$

  来找到最优切分变量和最优切分点(j^*,s^*)。
- 基于(j,s)，将输入空间划分成两个子区域，并对每个子区域重复上述划分过程，直到满足停止条件为止。

特征与切分点

- 例如考虑$\{((0.5,0)^T,1),((1,4)^T,3),((1.5,2)^T,7)\}$，每维特征都有两个切分点.

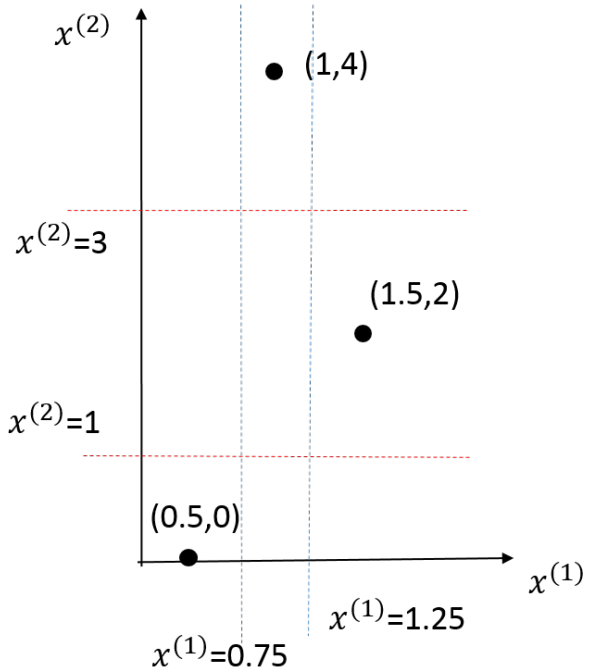

Figure: 特征的切分点

- 对$x^{(1)}$来说，共有两个切分点$x^{(1)}=0.75$和$x^{(1)}=1.25$，
  - 切分点$x^{(1)}=0.75$将上述三个样本切分成两个区域$R_1(1,0.75)$和$R_2(1,0.75)$，且

$$\begin{aligned}&\sum_{x_i\in R_1(1,0.75)}(y_i-\hat{c}_1)^2+\sum_{x_i\in R_2(1,0.75)}(y_i-\hat{c}_2)^2\\ =\;&(1-1)^2+[(3-\frac{3+7}{2})^2+(7-\frac{3+7}{2})^2]=8\end{aligned}$$

  - 切分点$x^{(1)}=1.25$将上述三个样本切分成两个区域$R_1(1,1.25)$和$R_2(1,1.25)$，且

$$\begin{aligned}&\sum_{x_i\in R_1(1,1.25)}(y_i-\hat{c}_1)^2+\sum_{x_i\in R_2(1,1.25)}(y_i-\hat{c}_2)^2\\ =\;&[(1-\frac{1+3}{2})^2+(3-\frac{1+3}{2})^2]+(7-7)^2=2\end{aligned}$$

- 对$x^{(1)}$来说，最优切分点为$x^{(1)}=1.25$.

- 对$x^{(2)}$来说，共有两个切分点$x^{(2)}=1$和$x^{(2)}=3$，
  - 切分点$x^{(2)}=1$将上述三个样本切分成两个区域$R_1(2,1)$和$R_2(2,1)$，且

$$\begin{aligned}&\sum_{x_i\in R_1(2,1)}(y_i-\hat{c}_1)^2+\sum_{x_i\in R_2(2,1)}(y_i-\hat{c}_2)^2\\ =\;&(1-1)^2+[(3-\frac{3+7}{2})^2+(7-\frac{3+7}{2})^2]=8\end{aligned}$$

  - 切分点$x^{(2)}=3$将上述三个样本切分成两个区域$R_1(2,3)$和$R_2(2,3)$，且

$$\begin{aligned}&\sum_{x_i\in R_2(2,3)}(y_i-\hat{c}_1)^2+\sum_{x_i\in R_2(2,3)}(y_i-\hat{c}_2)^2\\ =\;&[(1-\frac{1+7}{2})^2+(7-\frac{1+7}{2})^2]+(3-3)^2=18\end{aligned}$$

- 对$x^{(2)}$来说，最优切分点为$x^{(2)}=1$.
- 比较两个特征的最优切分点对应的平方误差和，显然$x^{(1)}$为当前最优切分变量，且最优切分点为$x^{(1)}=1.25$.

最小二乘回归树模型的剪枝算法

- 和分类任务的CART剪枝处理框架一致.
- 采用

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

来计算子树T的损失函数，其中

  - $C(T) = \sum_{t=1}^{|T|} N_t Q_t(T)$.
  - 叶结点t对应的单元R_t的预测误差$Q_t(T)$采用平方误差，即

$$Q_t(T) = \frac{1}{N_t} \sum_{x_i \in R_t} (y_i - \hat{c}_t)^2,$$

其中$\hat{c}_t$为R_t中x_i对应的y_i的均值，即$\hat{c}_t = \frac{1}{N_t} \sum_{x_i \in R_t} y_i$.

小结

- 决策树模型是基于特征对实例进行分类的树形结构.
- 从根结点到每个叶结点的路径对应于一条IF-THEN分类规则.
- 决策树的学习算法都采用贪心策略.
- ID3算法、 C4.5算法和 CART 算法分别采用信息增益最大准则、信息增益率最大准则和基尼指数最小准则来选择最优划分特征.
- ID3算法和C4.5算法对数据集进行多路划分，而CART算法则对当前数据集进行二分划分，前者产生的决策树层次较浅.
- CART算法也用于回归问题(最小二乘回归树模型).
- 为了降低过拟合风险，通常需要对决策树进行“预剪枝”或“后剪枝”处理.