机器学习笔记–决策树

空修菜

1 决策树（Decision trees）

在 SVM 模型中，我们使用了核方法 ϕ 进行映射而得到高维度的 feature。但决策树（Decision trees）不用类似的东西就可以得到一个非线性的假设函数。决策树就是将不能用线性模型进行估计的数据，进行二分的划分，直到找到合理的划分方式。

1.1 区域的选择（Selecting Regions）

在进行区域划分时，主要要找**叶节点**（leaf node）、**feature** 以及一个**threshold**。从根节点自顶向下递归地进行划分。给定一个输入空间或者数据集 $\mathcal{X}$, 将其二分为不同的区域 R_1, R_2, 通常被二分的区域称为**母区域**（parent region），新划分产生的新区域称为**子区域**（child region）。

给定母区域 R_p, 一个 feature 的索引值 j(X_j 就是向量 X 的第 j 个分量) 以及一个阀值 $t \in \mathbb{R}$, 可以得到两个不相交的子区域 R_1, R_2:

$$R_1 = \{X : X_j < t, X \in R_p\}$$
$$R_2 = \{X : X_j \geq t, X \in R_p\}.$$

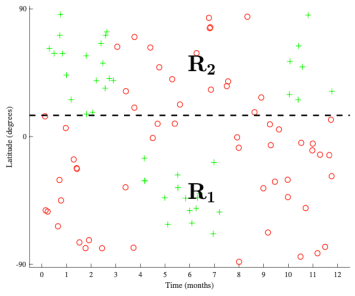

Figure 1.1: 整个数据集二分为两个区域 $\mathbf{R_1}$,$\mathbf{R_2}$

对新得到叶节点进行递归，将得到新的划分区域。如下图

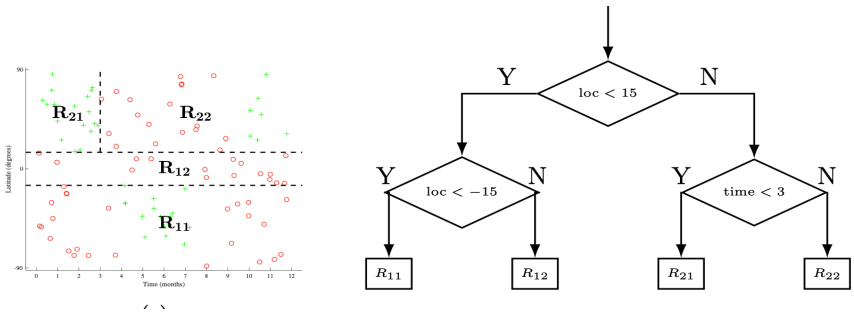

Figure 1.2: 对叶节点进行进一步的递归划分

1.2 定义损失函数（Define a Loss Function ）

选定某个 threshold 后, 就可以进行分类, 可以进行分类就需要判断分类的效果. 比如准确归类的概率或者错误归类的概率. 这就涉及到损失的概念. 本节关于损失有两概念，第一个是**错误归类损失函数**（misclassification loss）；第二个是**交叉熵**（cross entropy）。

1.2.1 Misclassification Loss

为方便说明, 考虑二分类问题. 给定一个母区域 R_p, 存在不同的 feature 以及阀值对 R_p 进行划分, 划分得到两个子区域 R_1, R_2，所以需要确定哪个划分效果更好，为此引入划分所得区域的损失函数 $L(R_i), i = p, 1, 2$. 对于母区域来说, 由于它本身就有两部分构成, 所以母区域本身已经有了一种划分. 划分所得的子区域的 Loss 表示加权均值:

$$\frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}.$$

其中，$|R_i|$ 表示区域 R_i 中的元素的总数. 在上式中.

将 $\mathcal{X}$ 视作一个母区域 R_p, 对于每一个支结构也可以进行这样的加权重的划分，这就得到每一层的损失 $L(R_{i\ldots j})$。如果是一个比较好的划分，递归得到的叶节点的损失函数较小；如果是一个不太好的分割，叶节点的损失就会大一些；由于 $L(R_p)$ 不变，那么式子

$$L(R_p) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}$$

比较大的时候，就说明划分的效果比较好；反之，则说明效果比较弱，要谨慎或者重新选择. 所以，由上面式子可知，要做的就是最大化上面的式子.

损失函数 L 可以定义为**错误归类损失**（misclassification loss）$L_{misclass}$:

$$L_{misclass}(R) = 1 - \max_c(\hat{p}_c),$$

其中, c 是对 R 的分类 $c = \{c_1, c_2\}$, $\hat{p}_c$ 就是属于 c 类的元素占整个 R 的比例. 比如:R 由 300 个正数和 500 个负数构成, 所以 R 自身就是一个二分类. c_1 类就是 R 中的 300 个正数, c_2 就是 R 中的 500 个负数, 此时 $\hat{p}_{c_1} = 300/(500 + 300) = 3/8$, 相应地, $c_2 = 5/8$, 所以 $L(R) = 1 - \max\{3/8, 5/8\} = 3/8$.

错误归类函数的一个缺点是它对归类的概率大变化并不敏感. 比如一个新得到的叶节点形成的类, 有可能在叶节点的类的概率 (占比) 改变后, 总的 $L_{misclass}$ 保持不变.

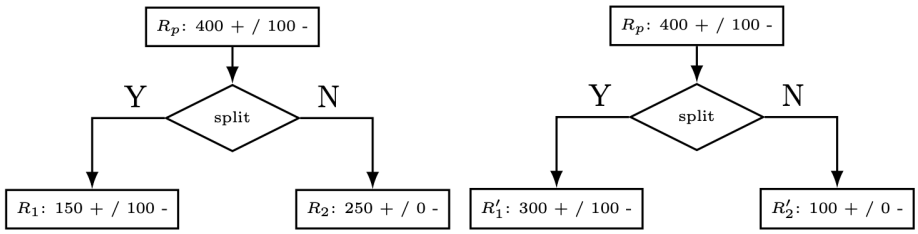

Figure 1.3: 同一个母区域 R_p 下, 不同分类方式的叶节点下, 不同的类的概率产生相同的错误归类概率.

上图中, 由 Loss 函数的定义, 我们可以得到母区域 R_p 以及每一个叶节点的错误损失:

$$\begin{aligned}
L(R_p) &= 1 - \max\{400/500, 100/500\} = 1/5 \\
L(R_1) &= 1 - \max\{150/250, 100/250\} = 2/5 \\
L(R_2) &= 1 - \max\{250/250, 0\} = 0 \\
L(R'_1) &= 1 - \max\{300/400, 100/400\} = 1/4 \\
L(R'_2) &= 1 - \max\{100/100, 0/100\} = 0.
\end{aligned}$$

此时可以发现类的概率是不同的 (此处,"**类**"指的是棱形框图中的判断条件形成的元素集合). 由前面的公式, 可以得到

$$\begin{aligned}
\frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|} &= \frac{1}{2} \times \frac{2}{5} + \frac{1}{2} \times 0 = \frac{1}{5} \\
\frac{|R'_1|L(R'_1) + |R'_2|L(R'_2)}{|R'_1| + |R'_2|} &= \frac{4}{5} \times \frac{1}{4} + \frac{1}{5} \times 0 = \frac{1}{5},
\end{aligned}$$

所以

$$L(R_p) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|} = L(R_p) - \frac{|R_1'|L(R_1') + |R_2'|L(R_2')}{|R_1'| + |R_2'|} = 0.$$

此时. 两个不同的划分的错误归类损失是是相等的, 叶节点的损失也是相等的. 这说明损失函数 L 对于类的概率变化并不敏感, 引进**交叉熵** (cross-entropy) 作为新的 loss function.

1.2.2 Cross Entropy

关于区域 R 的交叉熵 L_{cross} 定义为:

$$L_{cross}(R) = -\sum_c \hat{p}_c \log_2 \hat{p}_c$$

在考虑二分类问题时, 交叉熵以及损失函数就可以具体地写为:

$$L_{misclass}(R) = L_{misclass}(\hat{p}) = 1 - \max\{\hat{p}, 1-\hat{p}\}$$
$$L_{cross}(R) = L_{cross}(\hat{p}) = -\hat{p}\log\hat{p} - (1-\hat{p})\log(1-\hat{p}).$$

已知 $\hat{p} \in [0,1]$, 当 $\hat{p} \in [0, 1/2)$ 时,

$$L_{misclass}(\hat{p}) = 1 - \max\{\hat{p}, 1-\hat{p}\} = 1 - (1-\hat{p}) = \hat{p}.$$

当 $\hat{p} \in [1/2, 1]$,

$$L_{misclass}(\hat{p}) = 1 - \max\{\hat{p}, 1-\hat{p}\} = 1 - \hat{p}$$

将上面两个式子写到一起得到

$$L_{misclass}(\hat{p}) = \begin{cases} \hat{p} & 0 \le \hat{p} < 1/2 \\ 1-\hat{p} & 1/2 \le 0 \le 1 \end{cases}$$

同时, 容易证明 L_{cross} 是一个凹函数, 对其求导并令为 0, 将得到函数的极大值点.

$$0 = \frac{\partial L_{cross}}{\partial \hat{p}} = \log\frac{1}{\hat{p}} + \log(1-\hat{p})$$

所以, 当 $\hat{p} = 1/2$ 时, 函数 L_{cross} 取得极大值 $L_{cross}(1/2) = 1$.

1.3 feature 选择

设 X 是一个取有限个值的离散随机变量，其概率分布为

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

将 X 的熵 (Entropy) 定义为

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

由数学期望的定义，可以将上式写为

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i = -\sum_{i=1}^{n} P(X = x_i) \log_2 P(X = x_i) = \mathbb{E}[-\log_2 P(X)],$$

由以上定义来看，Entropy 求的是 $-\log P(X)$ 的期望, $H(X)$ 反映的实际上就是随机变量 X 的一个稳定程度。它的值越小，稳定程度就好；值越大，不确定性就越大。

当 X 只取两个值 1、2 时，Entropy 就是前面的 cross Entropy.

随机变量 (X, Y) 的联合概率分布为

$$P(X = x_i, Y = y_j) = p_{ij}, \quad i = 1, , 2, \ldots, n; \quad j = 1, 2, \ldots, m.$$

条件熵 (condition entropy)$H(Y|X)$ 是在 X 的条件下 Y 的不确定性。在 $X = x_i$ 时，由条件概率的定义，$Y = y_j$ 的条件概率可以写为

$$p(Y = y_j | X = x_i) = \frac{p(x_i, y_j)}{p(x_i)},$$

所以条件熵 $H(Y|X = x_i)$ 就可以按照 Entropy 定义写为

$$H(Y|X = x_i) = -\sum_{j=1}^{m} \frac{p(x_i, y_j)}{p(x_i)} \log_2 \frac{p(x_i, y_j)}{p(x_i)},$$

所以，条件熵 $H(Y|X)$ 就可以定义为

$$\begin{aligned} H(Y|X) &= \sum_{i=1}^{n} p(X = x_i) H(Y|X = x_i) \\ &= \sum_{i=1}^{n} p(x_i) \Big(-\sum_{j=1}^{m} \frac{p(x_i, y_j)}{p(x_i)} \log_2 \frac{p(x_i, y_j)}{p(x_i)} \Big) \\ &= -\sum_{i=1}^{n} \sum_{j=1}^{m} p(x_i, y_j) \log_2 \frac{p(x_i, y_j)}{p(x_i)}. \end{aligned}$$

Definition 1.1. *A 是一个给定的 feature，D 是一个数据集，information gain $g(D,A)$ 定义为 D 的 Entropy $H(D)$ 与 $D|A$ 的 Entropy 的差，即*

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

由于 $H(D)$ 在给定 D 后就固定下来，所以当 $H(D|A)$ 越小，那么 g 就越大. 同时，$H(D|A)$ 越小就表示按照该特征分类后的稳定性就越好，所以如果能够找到一个使得 g 最大的 feature$A$，那么这个 A 就是一个好的 feature. 所以 g 就确定了一种 feature 选择标准.

在使用信息增益 g 进行 feature 选择时，倾向于选择可取值较多的 feature，存在偏差问题. 弥补的办法是使如下定义的信息增益比.w

Definition 1.2. *A 是一个 feature，$H_A(D)$ 是按照 A 得出的 Entropy，*

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

1.4 决策树剪枝（Pruning of Tree）

决策树是对给定数据集 D 递归地生成的，因此对于 D 来说，所得的决策树 T 的分类是精确的. 但是要用该模型来进行预测效果就比较差，存在 overfitting 问题. 因此，要将 T 中的一些分支拿掉. 两个很自然的问题就产生了:(a) 什么时候该剪枝;(b) 依据什么标准剪枝?

关于 (a)，如果模型 T 关于 test set 的效果不好. 就该剪枝；关于 (b)，由于是预测，不准确的情况必然存在，对于每一个叶节点，如果去掉该点后，预测值和 target value 的距离为 0，或者降低了，该叶节点就可以作为剪枝点. 这就引入 loss function 的概念.

设决策树 T 的叶节点的个数为 $|T|$，t 是 T 的某个叶节点，t 的样本点个数为 N_t，其中 k 类的样本点有 N_{tk} 个，$k = 1, 2, \ldots, C$，$H_t(T)$ 是叶节点 t 的 Entropy，$\alpha \geq 0$ 是参数. loss function 定义为:

$$C_\alpha(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha|T|, \tag{1.1}$$

其中 $H_t(T)$ 定义为:

$$H_t(T) = -\sum_k \frac{N_{tk}}{N_t} \log \frac{N_{tk}}{N_t}.$$

在式子 (1.1) 中，第一部分表示模型对 train set 与模型的拟合程度，$|T|$ 表示树的复杂度. 在确定 α 后，当 T 的预测误差越小，说明分支就越多，所

以 T 的复杂度就越高; 反之, 当 T 较小, T 的叶节点就少, 预测误差就会大一些. (1.1) 式反映了两者的平衡.

1.5 分类回归树 (CART) 算法

CART(Classification And Regression Tree) 回归树和分类树, 既可以用于分类也可用于回归. CART 包括两个部分, 树的生成; 树的剪枝; 树的生成又可以进一步分为分类树生成和回归树生成.

1.5.1 回归树 (Regression Tree)

设 S 为训练集

$$S = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\}.$$

假设存在树 T 将 S 共划分为 M 类, 每一类记为 $R_i,\ i = 1, 2, \ldots, M$. 每一类 R_i 对应着一个值 c_i. 回归树就可以写为:

$$f(x) = \sum_{i=1}^{M} c_i 1\{x \in R_i\},$$

所以预测误差就可以写为

$$F(x) = F(x^{(1)}, x^{(2)}, \ldots, x^{(N)}) = \sum_{x^{(i)} \in R_i} \left(y^{(i)} - f(x^{(i)})\right)^2.$$

一般地, R_i 上所有样本点的 $y^{(i)}$ 的均值表示该类的代表值, 即

$$\hat{c}_{i_0} = \text{ave}\{y^{(i)} : x^{(i)} \in R_{i_0}\}.$$

在进行特征空间的划分时, 取变量 $x_{(i)}$ 的第 j 个分量 $x_j^{(i)}$, 并将该分量的值作为 threshold$s$, 所以有

$$R_1(j, s) = \{x^{(i)} : x_j^{(i)} \le s\}$$
$$R_2(j, s) = \{x^{(i)} : x_j^{(i)} > s\}$$

当 j 固定式，通过求解如下式子来得到 threshold s,

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

$$\hat{c_1} = \text{ave}\{y^{(i)} : x^{(i)} \in R_1(j,s)\}, \quad \hat{c_2} = \text{ave}\{y^{(i)} : x^{(i)} \in R_2(j,s)$$

. 第一次计算上面的式子，将得到 (j,s), 依次对 S 分为两个区域。之后，递归对每个区域进行操作就得到一个回归树。

1.5.2 分类树 (Classification Tree)

设 S 有 K 个类, 样本点属于第 k 类的概率为 p_k, 定义 Gini 系数为:

$$Gini(S) = \sum_{k=1}^{K} p_k(1-p_k).$$

假设有训练集 D, 且有 C_k 个分类, $k = 1, 2, \ldots, K$. 由上面的定义, 可以得到 D 的 Gini 系数

$$\begin{aligned} Gini(D) &= \sum_{k=1}^{K} \frac{|C_k|}{|D|}\Big(1 - \frac{|C_k|}{|D|}\Big) \\ &= \sum_{k=1}^{K} \frac{|C_k|}{|D|} - \sum_{k=1}^{K} \Big(\frac{|C_k|}{|D|}\Big)^2 \\ &= 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|}\right)^2. \end{aligned}$$

当选择了某个 feature$A$ 时，根据 A 就可以将 D 分为两个部分 D_1, D_2, 在 A 条件下，D 的 Gini 系数 $Gini(D|A)$ 可以写为

$$Gini(D|A) = \sum_{k=1}^{2} \frac{|D_i|}{|D|} Gini(D_i),$$

Gini 系数像 Entropy 一样，反映了一种不确定性、不稳定性，值越大越不稳定。

至此,关于 feature 的选择,已经有了三个指标:(a)$L_{misclass}(p)$; (b)Entropy $H(p)$; (c)Gini coefficient $Gini(p)$. 在种类 $K = 2$ 时，分别将三个函数的表达式写出:

$$\begin{aligned} L_{misclass}(p) &= \begin{cases} p & 0 \le p < 1/2 \\ 1-p & 1/2 \le 0 \le 1 \end{cases} \\ H(p) &= -p\log_2 p - (1-p)\log_2(1-p) \\ Gini(p) &= 2p(1-p). \end{aligned}$$

为便于比较，将三个函数的图像画在一起就得到如下的图像, 其中 log 以 2 为底数、$H(p)/2$:

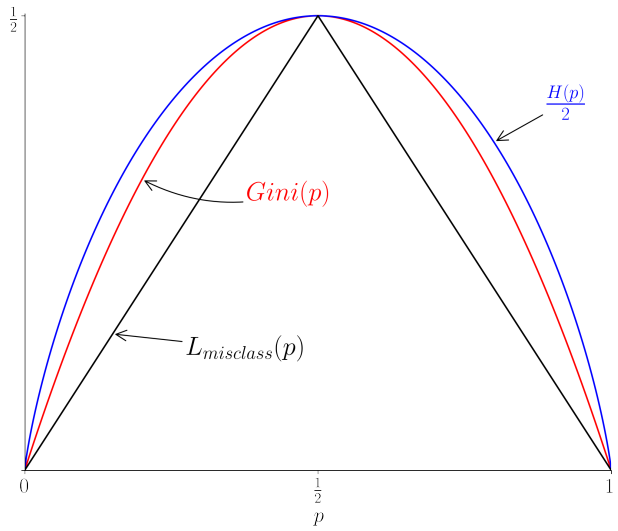

Figure 1.4: 该图的 log 的底数为 2,$H(p)$ 取其一半. 在 $K = 2$ 时, 最稳定的是 L, 最不稳定的 H, 但是 L 对比例的变化并不敏感，相比之下 $Gini$ 是一个较好的折中选择.

1.5.3 剪枝（Pruning）

对所得到的树 T 从叶节点到最后的根节点进行逐步剪枝，得到一个子树序列 $\{T_0, T_1, \ldots, T_n\}$, 之后利用交叉验证法，从中找到最后的树作为回归树. CART 算法所使用的剪枝策略是 CCP(Cost Complex Prune) 策略.

1.6 Summary of Decision Tree

1. 决策树 T 的建立有三种算法.TD3 算法; C4.5 算法; CART 算法.
2. TD3 算法和 C4.5 算法的生成树:
   - TD3 算法选择特征和 threshold 的方法是计算并选取最大的信息增益 (information gain).
   - C4.5 算法通过信息增益比来确定特征和 threshold.
3. CART 算法:
   - CART 算法可以生成回归树, 也可以生成分类树.

- 选择特征和 threshold 的方法是 Gini 系数.

4. 剪枝. 剪枝分为预剪枝与后剪枝 (post pruning).

   - 预剪枝是在生成树 T 的时候, 也对树 T 进行剪枝 (控制). 常见的剪枝方式有:
     (a). 限制树 T 的最大深度;
     (b). 限制每个节点的叶节点数;
     (c). 限制每个节点要包含的数据数.
   - 后剪枝的算法有:
     (a). 错误率降低剪枝 (Reduced Error Pruning);
     (b). 悲观误差剪枝 (Pessimistic Error Pruning);
     (c). 最小误差剪枝 (Minimum Error Pruning);
     (d). 代价复杂剪枝 (Cost-Complexity Pruning). 这是 CART 所使用的剪枝方式.

5. 决策树的偏差和方差:

   - 若不限制决策树 T 的生长, 则 T 对训练集有较好的效果, 但对验证集的效果差. 即方差高, 偏差低, 过拟合. 这可以解释为: 使用验证集时, 模型又得到了新的数据, 又得到了新的映射规律, 且与原来的映射规律相差较大.
   - 解决偏差与方差矛盾的模型是集成学习模型.