Lecture11 决策树&朴素贝叶斯

1. 决策树 Decision Tree

介绍

- 决策树是一个很流行的**分类**方法
- 易于理解，算法方法简单
- 没有线性的假设

历史

- CART（Classification And Regression Trees）：Friedman 1977
- ID3 和 C4.5 家族：Quilan 1979-1983
- 在 1990 中期提出了一些优化方法（剪枝，数值特征的决策树划分等）

应用

- 植物学
  - 不列颠群岛新植物群
- 药物搜索
  - 印度糖尿病诊断
  - 早期诊断为急性心肌梗死
- 计算生物学
  - 基因交互作用

树分类器

决策树是从根向下生长的，我们的想法是利用**信息熵 information entropy** 的概念，将这些例子沿着树向下发送

通常构建一个决策树的方法

1. 从包含所有样本的根节点开始
2. 贪心的选择最好的特征来构建分支，划分的标准是**节点的纯度 node purity**
3. 叶子节点代表分类结果

任务

给定：训练数据 $(x_1, y_1), \dots (x_n, y_n)$

- $x_i \in \mathbb{R}^d$
- y_i 是离散的（可分类的）$y_i \in Y$（例如 $Y = \{-1, +1\}, Y = \{0, 1\}$）

任务：学习一个分类函数 $f : \mathbb{R}^d \to Y$

在树分类器中

- 不需要将类型特点转换为数值的特点

- 模型是一棵树

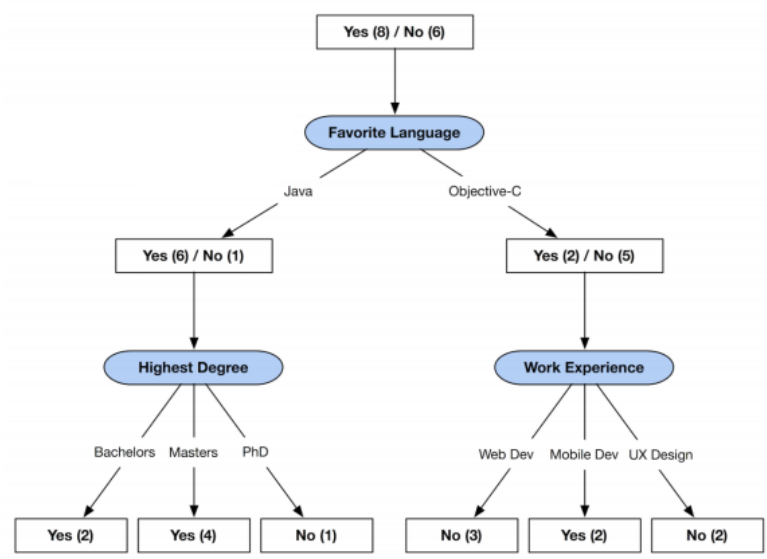
示例

我们想要招募一些程序员，他们有不同的特点，我们想要通过构建一个决策树来判断是否应该招收某个员工

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Bachelors	Mobile Dev	Objective-C	TRUE	yes
Masters	Web Dev	Java	FALSE	yes
Masters	Mobile Dev	Java	TRUE	yes
PhD	Mobile Dev	Objective-C	TRUE	yes
PhD	Web Dev	Objective-C	TRUE	no
Bachelors	UX Design	Objective-C	TRUE	no
Bachelors	Mobile Dev	Java	FALSE	yes
PhD	Web Dev	Objective-C	FALSE	no
Bachelors	UX Design	Java	FALSE	yes
Masters	UX Design	Objective-C	TRUE	no
Masters	UX Design	Java	FALSE	yes
PhD	Mobile Dev	Java	FALSE	no
Masters	Mobile Dev	Java	TRUE	yes
Bachelors	Web Dev	Objective-C	FALSE	no

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Masters	UX Design	Java	TRUE	?

转换为决策树如下



- 非叶节点为选择分类的属性
- 分支为属性的不同的值
- 叶子节点为划分的类型

划分标准 - ID3

信息熵 Information Entropy

核心的思想是要选择一个进行划分的属性

我们希望有一些评判指标来衡量节点中实例的**同质性 homogeneity** 或**不纯性 impurity**

- 量化每个节点上不同类的混合程度
- 如果划分后的有很多样本属于不同类，则混乱程度增加
- 如果划分后所有样本属于同一类，则混乱程度减少

一个很好的常用的划分标准是**信息熵 Information Entropy**

对于**二分类问题**来说：

$$Entropy(S) = -p_{\oplus}log_2p_{\oplus} - p_{\ominus}log_2p_{\ominus}$$

- $p_{\oplus}$：划分为**正类**的样本的**占比**
- $p_{\ominus}$：划分为**负类**的样本的**占比**
- $Entropy(s)$：信息熵，描述状态的混乱程度

如果是**多分类**的计算，对于有 c 个类型的情况来说：

$$Entropy(S) = \sum_{i=1}^{c} -p_ilog_2p_i$$

信息增益 Information Gain

现在，对于每一个节点来说，它都有一个描述划分出来的纯度的效果

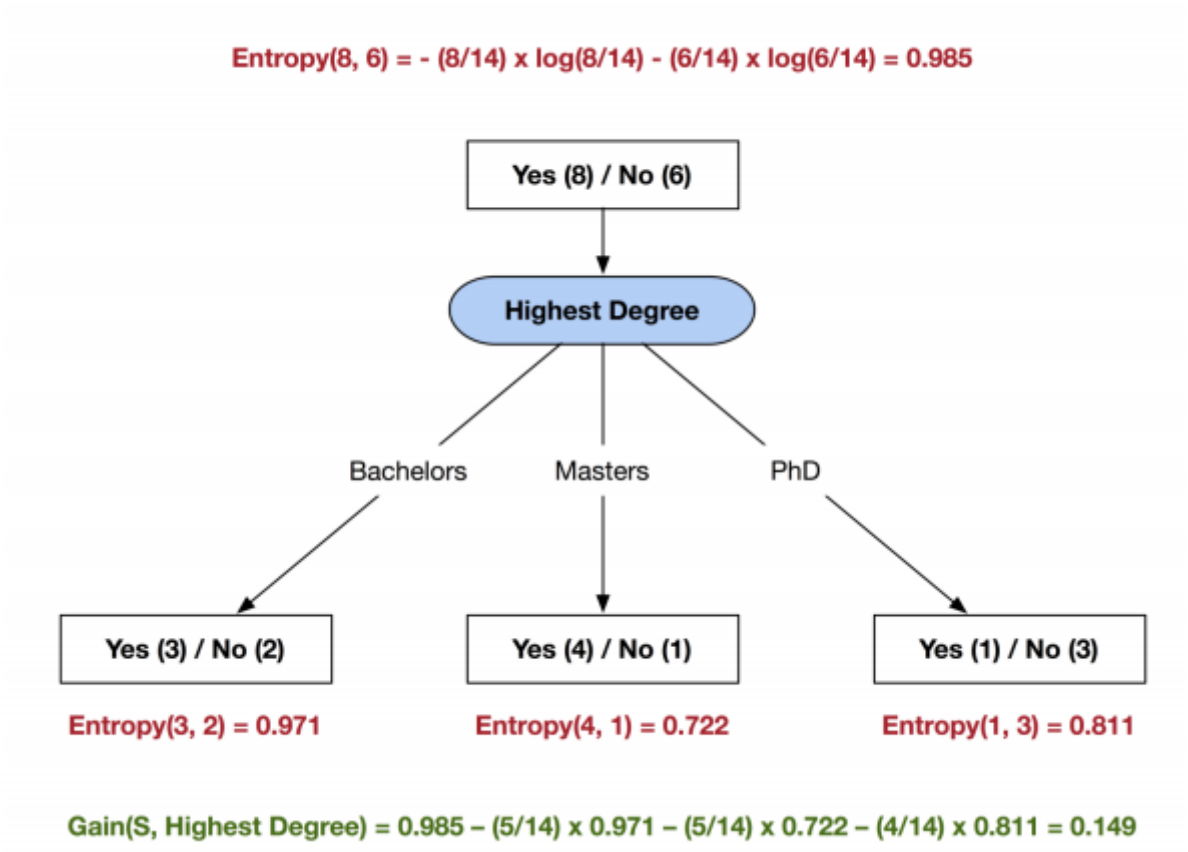
如何根据信息熵确定哪一个属性是最好的划分标准呢？

我们使用**信息增益 Information Gain** 来度量根据属性划分样本后所导致的熵增的一种加权版

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- S：原始样本集
- A：属性集合
- v：某一个划分属性
- S_v：按照属性 v 划分后的样本集

以上面的示例为例



在没有按照 Highest Degree 划分前，样本集中有 8 个正样本，6 个负样本

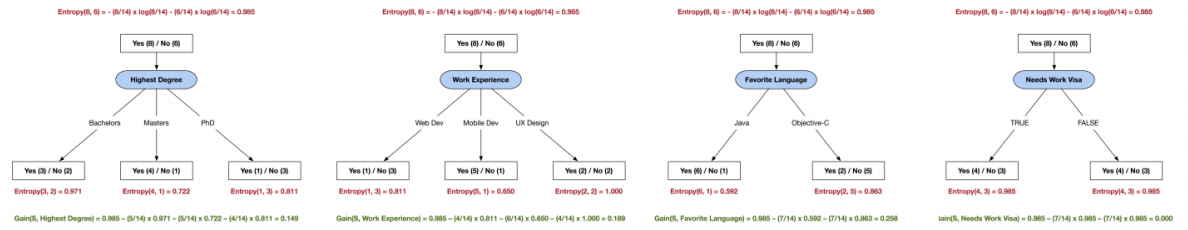
$$Entropy(S) = -\frac{8}{14} * log_2\frac{8}{14} - \frac{6}{14} * log_2\frac{6}{14} = 0.985$$

按照 Highest Degree 划分出了三个子样本集，它们的信息熵分别为

$$Entropy(S_{v1}) = 0.971$$
$$Entropy(S_{v2}) = 0.722$$
$$Entropy(S_{v3}) = 0.811$$

则信息增益为

$$Gain(S, Highest\ Degree) = 0.985 - \frac{5}{14}\times 0.971 - \frac{5}{14}\times 0.722 - \frac{4}{14}\times 0.811 = 0.149$$



- 分别计算四个属性的信息增益，可以发现，按照 Favourite Language 划分后，信息增益最大，所以优先选择它作为划分依据

划分好后，我们对**每个子节点递归采用相同的方法**

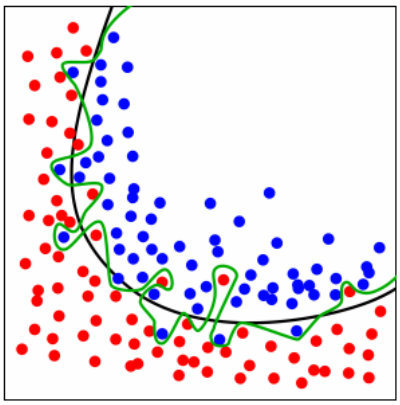
划分标准 - CART

- 采用一些贪婪的，自顶向下的算法
- 做**二路划分**，而不是多路划分
- 使用**基尼指数 Gini Index** 而不是信息熵

$$Gini = 1 - p_{\oplus}^2 - p_{\ominus}^2$$

过拟合与剪枝策略

如果划分足够细致的话，每个样本都能被单独划分到一个分支上，决策树的训练容易**过拟合**！



- 如图，可能黑色的线是一个比较好的划分标准，然而决策树可能过拟合到了绿色的线的样子

首先，决策树模型的复杂度可以大致由树的层数定义，树的层数越多，模型越复杂

为了避免决策树过拟合，可以采用两种方法

- **尽早停止决策树的生长**：在它达到完美类训练示例的点之前，停止生长决策树（很难知道何时停止）
- **先生成一个完全生长的决策树，然后进行剪枝**：目前最好的手段
  - 使用**验证集**来测试，如果删除某些节点，对决策树的影响是否比较大
  - 如果新树的性能不差于原始树，则删除一个子树

实践考虑

- 考虑提前进行**降维**以保留最具识别力的特征
- 使用**集成方法**（如随机森林）来有一个良好的表现
- 在训练之前**平衡数据集**，以防止树创建偏向于占优势的类的树，不同分类的样本的个数差异比较大，可能类型 A 的样本有100 个而类型 B 的样本只有 10 个
  - **Under-Sampling**：把类型多的样本减掉一些
  - **Over-Sampling**：人工合成一些样本数量较少的类型（例如，使用 SMOTE）

优缺点

优点

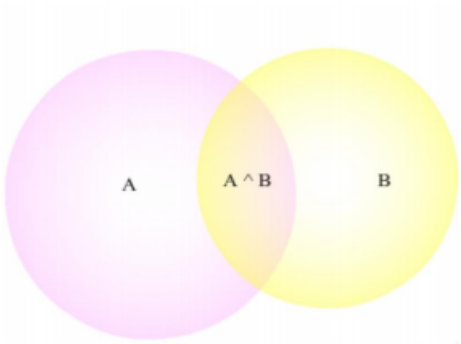
- 符合知觉，可解释
- 可以变成规则
- 非常适合分类数据
- 很容易去构建
- 不需要缩放数据

缺点

- 不稳定（样本的更改可能导致生成不同的树）
- 单变量（一次只拆分一个属性，不合并多个属性）
- 某个节点上的选择取决于前面的选择
- 需要平衡数据集

2. 朴素贝叶斯 Naive Bayes

条件概率



$$p(A \mid B) = \frac{p(A \wedge B)}{p(B)}$$
$$p(A \wedge B) = p(A \mid B) * p(B)$$

$p(A \wedge B)$ 可以被写成两种不同的表达

- $p(A \wedge B) = p(A \mid B) * p(B)$
- $p(A \wedge B) = p(B \mid A) * p(A)$

贝叶斯定律

那么，得到最经典的贝叶斯定理

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B)}$$

它们有不同的称呼

- posterior: $p(A|B)$
- prior: $p(A)$
- evidence: $p(B)$
- likelihood: $p(B|A)$

	A	!A	Sum
B	$p(A \wedge B)$	$p(\neg A \wedge B)$	$p(B)$
!B	$p(A \wedge \neg B)$	$p(\neg A \wedge \neg B)$	$P(\neg B)$
	$p(A)$	$p(\neg A)$	1

贝叶斯定理的另一种表现形式

$$p(A \mid B) = \frac{p(B \mid A) * p(A)}{p(B \mid A) * p(A) + p(B \mid \neg A) * p(\neg A)}$$

示例

现在举一个具体的示例

- A：病人有癌症
- B：病人的化验呈阳性

$$\begin{array}{ll} p(A) = 0.008 & p(\neg A) = 0.992 \\ p(B \mid A) = 0.98 & p(\neg B \mid A) = 0.02 \\ p(B \mid \neg A) = 0.03 & p(\neg B \mid \neg A) = 0.97 \end{array}$$

代入求得

$$p(A \mid B) = \frac{0.98 \times 0.008}{0.98 \times 0.008 + 0.03 \times 0.992} = 0.21$$

- 即在病人化验呈阳性的情况下，病人有癌症的概率仅为 0.21

朴素贝叶斯在机器学习的应用

我们为什么要引入贝叶斯框架？回想分类的框架

给定：训练数据 $(x_1, y_1), \ldots (x_n, y_n)$

- $x_i \in \mathbb{R}^d, y \in Y$

任务：学习一个分类函数 $f : \mathbb{R}^d \to Y$

现在我们要把 x 映射成 y，此时可以表示成

$$f(x) = p(y|x)$$

核心思想：我们通过建模 $p(y|x)$，即在给定 x，求 y 的分布，这样也就映射成了 $f(x) = y$

- 在辨别算法中，找到一个决策边界，将阳性样本和阴性样本分开
- 我们知道 $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$
- 为了进行预测，我们要使得 $\begin{aligned} \operatorname{argmax}_y p(y \mid x) &= \operatorname{argmax}_y \frac{p(x \mid y)p(y)}{p(x)} \\ \operatorname{argmax}_y p(y \mid x) &\approx \operatorname{argmax}_y p(x \mid y)p(y) \end{aligned}$

朴素贝叶斯分类器

介绍

- 一种概率模型
- 高可用的方法
- 应用程序域到自然语言文本文档
- 是朴素的，因为它做出了独立性的假设
- 简单的模型
- 在某些情况下，朴素贝叶斯模型可以与决策树和神经网络相媲美

模型构建

- 给定一组训练数据 (x_i, y_i)，其中 x_i 是特征向量，y_i 是离散的标签
- d 维的特征，以及 n 个样本
- 示例：考虑对文档进行分配，每个样本都是一个文档，一个文档中的特征可以是表示文档某个特定的单词存在或者不存在，标签是对文档的分类
- 我们有一个新的样本，它的特征向量 $x_{new} = (a_1, a_2, \ldots, a_d)$，我们想知道对于这个新的样本的标签 y_{new}

我们想找到

$$y_{\text{new}} = \operatorname{argmax}_{y \in \mathbb{Y}} \quad p\left(y \mid a_1, a_2, \cdots, a_d\right)$$

使用贝叶斯定理，即我们想获得

$$\begin{aligned} y_{\text{new}} &= \operatorname{argmax}_{y \in \mathbb{Y}} \frac{p\left(a_1, a_2, \cdots, a_d \mid y\right) * p(y)}{p\left(a_1, a_2, \cdots, a_d\right)} \\ y_{\text{new}} &= \operatorname{argmax}_{y \in \mathbb{Y}} \quad p\left(a_1, a_2, \cdots, a_d \mid y\right) * p(y) \end{aligned}$$

- 我们能够从训练数据中获得 $p\left(a_1, a_2, \cdots, a_d \mid y\right)$ 和 $p(y)$ 吗
  - $p(y)$ 可以很容易获得，因为只需要找标点是 y 的在样本中的占比
  - $p(a_1, a_2, \ldots, a_d|y)$ 不是很容易估计，除非我们有一个很大的样本集（这是一个统计概率）

所以，朴素贝叶斯假设特征向量的每一个维度对于标签都**条件独立**的

那么，给定样本的标签，满足

$$p\left(a_1, a_2, \cdots, a_d \mid y\right)=\prod_j p\left(a_j \mid y\right)$$

朴素贝叶斯分类找到某个特定的标签，最大化下面的式子

$$y_{\text{new}}=\operatorname{argmax}_{y \in \mathbb{Y}} \quad p(y) \prod_j p\left(a_j \mid y\right)$$

- 这时候要想找到每一个维度的概率 $p(a_j|y)$ 就比较容易了

算法

根据数据集学习概率

- 估计所有的 $p(y), \forall y \in Y$
- 估计所有的 $p(a_j|y), \forall y \in Y, \forall a_j$

分类，对于新的样本，选择最好的哪个标签，满足最大化下面的概率的那个标签

$$y_{\text{new}}=\operatorname{argmax}_{y \in \mathbb{Y}} \quad p(y) \prod_j p\left(a_j \mid y\right)$$

- 没有模型本身或超平面，只是计算训练示例中各种数据组合的频率

估计概率

如果数据集中某一个条件概率 $p(a_j|y)=0$，这其实是一个不好的情况，这时候我们要对**概率进行修正**

$$p(a_j|y)=\frac{n_c+1}{n_y+k_j}$$

- n_y：训练集中样本的 label 是 y 的个数
- n_c：训练集中样本的 label 是 y 且样本的特征 $x_j=a_j$ 的个数
- k_j：特征 x_j 的划分的值的个数

示例

数据集仍然是这个

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Bachelors	Mobile Dev	Objective-C	TRUE	yes
Masters	Web Dev	Java	FALSE	yes
Masters	Mobile Dev	Java	TRUE	yes
PhD	Mobile Dev	Objective-C	TRUE	yes
PhD	Web Dev	Objective-C	TRUE	no
Bachelors	UX Design	Objective-C	TRUE	no
Bachelors	Mobile Dev	Java	FALSE	yes
PhD	Web Dev	Objective-C	FALSE	no
Bachelors	UX Design	Java	FALSE	yes
Masters	UX Design	Objective-C	TRUE	no
Masters	UX Design	Java	FALSE	yes
PhD	Mobile Dev	Java	FALSE	no
Masters	Mobile Dev	Java	TRUE	yes
Bachelors	Web Dev	Objective-C	FALSE	no

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Masters	UX Design	Java	TRUE	?

首先估算出 $p(yes) = \frac{8}{14} = 0.572$，$p(no) = \frac{6}{14} = 0.428$

条件概率如下

$$
\begin{array}{ll}
p(\text{ masters } \mid \text{ yes }) = 4/8 & p(\text{ masters } \mid \text{ no }) = 1/6 \\
p(UX \text{ Design } \mid \text{ yes }) = 2/8 & p(\text{ UX Design } \mid \text{ no }) = 2/6 \\
p(\text{ Java } \mid \text{ yes }) = 6/8 & p(\text{ Java } \mid \text{ no }) = 1/6 \\
p(\text{ TRUE } \mid \text{ yes }) = 4/8 & p(\text{ TRUE } \mid \text{ no }) = 3/6
\end{array}
$$

预测 $y = yes$ 的概率为

$$p(yes) * p(Master|yes) * p(UX\ \ Design|yes) * p(Java|yes) * p(TRUE|yes) = 0.026$$

预测 $y = no$ 的概率为

$$p(no) * p(Master|no) * p(UX\ \ Design|no) * p(Java|no) * p(TRUE|no) = 0.002$$

所以 $y_{new} = yes$

比如说，如果要对 $p(master|yes)$ 进行修正，观察到特征 Highest Degree 一共有 3 个取值（Bachelors, Masters, PhD）

所以说，修正后

$$p(master|yes) = \frac{4+1}{8+3} = \frac{5}{11}$$

应用

文档分类器

- 学习感兴趣的新闻文章
- 学习根据话题分类网页
- 从非垃圾邮件分类垃圾邮件
- 朴素贝叶斯是最有效的算法之一
- 我们应该使用什么属性来表示文本文档

总结

- 贝叶斯是一个线性分类器
- 它很简单也很容易实现
- 它对于文本分类效果很好