

朴素贝叶斯（native Bayes）

$$P(Y|X) = \frac{P(X, Y)}{P(X)} = \frac{P(Y)P(X|Y)}{\sum_Y P(Y)P(X|Y)}$$

将输入 x 分到后验概率最大的类 y

$$y = \operatorname{argmax}_{c_k} P(Y = c_k) \prod_{j=1}^n P(X_j = x^{(j)} | Y = c_k)$$

注意点：

1 朴素贝叶斯对条件概率分布做了条件独立性的假设。

$$P(X = x | Y = c_k) = P(X^{(1)} = x^{(1)}, \dots, X^{(n)} = x^{(n)} | Y = c_k)$$

$$= \prod_{j=1}^n P(X^{(j)} = x^{(j)} | Y = c_k)$$

相关知识：

1 先验概率与后验概率

事情还没有发生,要求这件事情发生的可能性的的大小,是先验概率.

事情已经发生,要求这件事情发生的原因是由某个因素引起的可能性的大小,是后验概率.

先验概率是指根据以往经验和分析得到的概率,如全概率公式,它往往作为“由因求果”问题中的“因”出现。后验概率是指在得到“结果”的信息后重新修正的概率,如贝叶斯公式中的,是“执果寻因”问题中的“因”。先验概率与后验概率有不可分割的联系,后验概率的计算要以先验概率为基础。

2 利用极大似然估计计算的时候可能出现概率值为 0 的情况,会影响到后验概率的结果,使分类产生偏差。因此采取增加 λ 的方式,称为拉普拉斯平滑。

优点：

- 1 因为独立性的假设,模型包含的条件概率的数量大为减少,因此高效,且易于实现。
- 2 对小规模的数据表现很好,适合多分类任务,适合增量式训练。

缺点：

- 1 因为独立性的假设,牺牲了一定的分类准确率,分类的性能不一定高。

2 对输入数据的表达形式很敏感。

KNN(k-nearest neighbor)

注意点：

1 K 值的减小意味着整体模型变得复杂，容易发生过拟合。K 值变大意味着整体模型变得简单。K 值通常选取一个比较小的数据，采用交叉验证法来选取最优的 K 值。常用的分类决策规则是多数表决，对应于经验风险最小化。

2 KNN 的基本做法是：对给定的训练实例点和输入实例点，首先确定输入实例点的 k 个最近邻训练实例点，然后利用这 k 个训练实例点的类的多数来预测输入实例点的类。

3 KNN 模型对应于基于训练数据集对特征空间的一个划分。KNN 中，当训练集，距离度量，k 值和分类决策规则确定后，其结果唯一确定。

4 KNN 的实现需要考虑如何快速搜索 k 个最近邻点。kd 树是一种便于对 k 维空间中的数据进行快速检索的数据结构。kd 树是二叉树，表示对 k 维空间的一个划分，其每个节点对应于 k 维空间划分的一个超矩形区域。利用 kd 树可以省去对大部分数据点的搜索，从而减少搜索的计算量。kd 树的平均计算复杂度是 $O(\log N)$ 。

相关知识点：

优点：

- 1 思想简单，理论成熟，既可以用来做分类也可以用来做回归。
- 2 可用于非线性分类。
- 3 训练时间复杂度为 $O(n)$ 。
- 4 准确度高，对数据没有假设，对 outlier 不敏感。

缺点：

- 1 计算量大
- 2 样本不平衡（即有些类别的样本数量很多，而其他样本的数量很少）
- 3 需要大量的内存

决策树（decision tree）

熵（entropy）的定义：表示随机变量不确定性的度量。熵越大，随机变量的不确定性越大。

$$P(X = x_i) = p_i$$

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

$$H(Y|X) = \sum_{i=1}^n p_i H(Y|X = x_i)$$

如果对数以 2 为底，那么熵的单位是比特（bit），如果以 e 为底，那么单位是纳特（nat）。

信息增益（information gain）表示得知特征 A 的信息而使得训练数据集 D 的信息的不确定性减少的程度。信息增益大的特征具有更强的分类能力。

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

注意点：

1 因为从可能的决策树中直接选取最有决策树是 NP 完全问题。现实中采用启发式方法学习次优的决策树。决策树的学习算法包含 3 部分，特征性选择，树的生成和树的剪枝、常用的算法有 ID3，C4.5 和 CART。

2 特征选择的目的在于选取对训练数据能够分类的特征。特征选择的关键是其准则。常用的准则如下：

（1）样本集合 D 对特征 A 的信息增益（ID3）

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

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

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

其中，H(D)是数据集 D 的熵，H(D_i)是数据集 D_i 的熵，H(D|A)是数据集 D 对特征 A 的条件熵。D_i 是 D 中特征 A 取第 i 个值的样本子集，C_k 是 D 中属于第 k 类的样本子集。|D| 表示其样本容量，即样本个数。n 是特征 A 取值的个数，K 是类的个数。

（2）样本集合 D 对特征 A 的信息增益比（C4.5）

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

其中，g(D,A)是信息增益，H(D)是数据集 D 的熵。

（3）样本集合 D 的基尼指数（CART） CART 的决策树是二叉树

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

特征 A 条件下集合 D 的基尼指数：

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

3 决策树的生成。通常使用信息增益最大，信息增益比最大或者基尼指数最小作为特征选择的准则。决策树的生成往往通过计算信息增益或其他指标，从根节点

开始，递归地产生决策树。这相当于用信息增益或其他准则不断的选取局部最优的特征，或将训练集分割为能够基本正确分类的子集。

4 决策树的剪枝。由于生成的决策树存在过拟合的问题，需要对它进行剪枝，以简化学到的决策树。决策树的剪枝，往往从已生成的树上减掉一些叶节点或叶节点以上的子树，并将其父节点或根节点作为新的叶节点，从而简化生成的决策树。决策树的减枝分为预剪枝和后减枝，预剪枝通过验证集的方式实现，后减枝可以通过验证集或者损失函数来实现。

相关知识点：

优点：

1 计算量简单，可解释性强，比较适合处理有缺失属性值的样本，能够处理不相关的特征。

缺点：

1 容易过拟合（后续出现了随机森林，减小了过拟合的现象）

逻辑回归

线性回归

支持向量机（SVM）

神经网络

GBDT 和 XGBOOST

随机森林

kmeans

EM