# **常用算法简介**

接下来我们将介绍八个常用的机器学习算法。

## C4.5

C4.5作为一种决策树算法，前身是著名的ID3算法，二者的区别在于划分属性的准则，我们稍后介绍。

决策树是一类常用的机器学习算法，顾名思义，决策树是基于树结构进行决策的，这一点恰是人类面对决策问题的处理机制。算法通过把实例从根结点排列到某个叶子结点实现分类，其核心问题在于选择分裂属性和决策树的剪枝。常用的算法有ID3、C4.5、CART等，这些算法均采用自顶而下的贪婪算法，每个结点选择分类效果最好的属性将结点分裂成2个或多个子结点，下图为根据天气情况判断“周六是否适合打网球”的决策树模型。

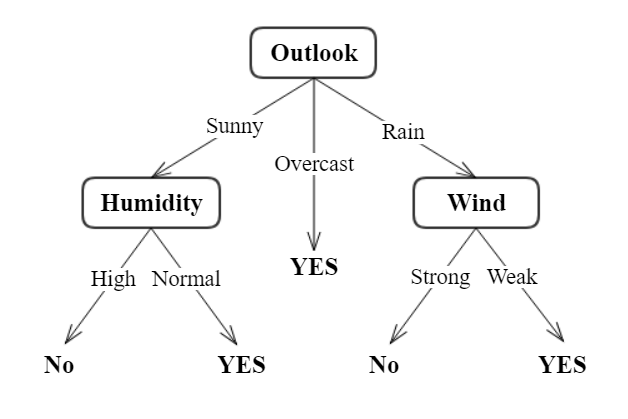


图3 决策树示例图

**Fig.**3 **Decision tree example diagram**

一般的，一棵决策树包含一个根结点、若干个内部结点和若干个叶结点，叶结点对应着决策结果，其他每个结点对应一个属性测试。由根结点到叶结点的过程即为决策过程，决策树学习的目的是为产生一种泛化能力较强的分类模型。通常先选取一个根结点，根绝其属性生成各个分支，再以每个分支为根结点或叶结点进一步划分，具体流程如下

|  |
| --- |
| **输入**：训练集  属性集  **过程**：函数TreeGenerate  1：生成结点node；  2：**if** *D*中样本全属于同一类别C **then**  3: 将node标记为*C*类叶结点；**return**  4：**end if**  5：**if**  **OR** *D*中样本在*A*上取值相同**then**  6： 将node标记为叶结点，其类别标记为*D*中样本数最多的类；**return**  7：**end if**  8：从*A*中选择最优划分属性；  9：**for** 的每一个值**do**  10： 为node生成一个分支；令表示中在上取值为的样本子集；  11： **if** 为空 **then**  12： 将分支结点标记为叶结点，其类别标记为中样本最多的类；**return**  13： **else**  14： 以TreeGenerate  15： **end if**  16：**end for**  **输出：**以node为根结点的一棵决策树 |

图4 决策树学习基本算法

**Fig.**4 **Basic learning algorithm of decision tree**

决策树的生成是一个递归过程，有三种情形会导致递归返回：①当前结点包含的样本全属于同一类别，无需划分；②当前属性即为空，或是所有样本在所有属性上的取值相同，无法划分；③当前结点包含的样本集合为空，不能划分。

对于第②种，直接将当前结点标记为叶结点，且类别设定为该结点所包含样本最多的类别；对于第③种，将当前结点标记为叶结点，将其类别设定为其父结点所包含样本最多的类别。

算法最核心的部分为第8行选择最优划分属性，不同的划分方法对应着不同的决策树算法。

我们先给出“信息熵”的定义：

假定当前样本集合*D*中第*k*类样本所占的比例为，则*D*的信息熵定义为

若使用某离散属性*a*对样本集*D*进行划分，设其有*V*个可能的取值，则会产生*V*个分支结点，第*v*个分支结点包含了*D*中所有属性*a*取值为的样本，即为。由此可计算的信息熵，考虑各分支结点样本数量不同，赋予第*v*个分支结点权重，于是可计算用属性*a*对样本进行划分所得到的“信息增益”如下

若在算法流程第8行选取使信息增益最大的属性作为最优划分属性，即可得到ID3决策树学习算法。

由于信息增益偏好取之数目较多的属性，因此可考虑采用“增益率”来选择最优划分属性，以减少此偏好带来的不利影响。增益率定义如下

其中

称为属性*a*的“固有值”

C4.5算法不直接采用增益率最大的划分属性，而是从候选划分属性中找出信息增益高于平均水平的属性，再从中选择增益率最高的。

## *k*-Means

在先前介绍的“无监督学习”方法中，研究最多、应用最广的一大领域是“聚类”。聚类试图将数据集划分为若干个互不相交的子集，每个子集称为一个“簇”。聚类算法可以解释数据的内在性质及规律，获取数据的一些潜在的概念，需注意这样的概念是事先未知的，在聚类完成后需要由使用者把握及解释。

通俗地讲，聚类是将相似的事物聚集在一起，而将不相似的事物划分到不同的类别的过程。那么，如何衡量两个事物之间的相似度，我们可通过定义“样本间距离”及“类间距离”进行衡量。

首先给定两样本与，常用的“距离度量”有

1. **欧氏距离**
2. **曼哈顿距离**
3. **切比雪夫距离**
4. **闵可夫斯基距离**

* 时，是曼哈顿距离；
* 时，是欧式距离；
* 时，是切比雪夫距离.

1. **马氏距离**

有*M*个样本向量，协方差矩阵记为，均值记为向量，则其中样本向量到的马氏距离表示为：

1. **汉明距离**

两个等长字符串与之间的汉明距离定义为两个（相同长度）字对应位不同的数量。例如字符串“1111”与“1001”之间的汉明距离为2。

1. **Pearson相关系数**
2. **夹角余弦**

需要说明的是，用距离作为亲疏程度的度量值时，距离越小，样品之间的关联性越大；用相似系数作为亲疏程度的度量值时，相似系数的绝对值越大，意味着指标之间的关联性越大。

*k-*Means算法需事先给定聚类个数*k*，聚类目标为最小化*m*个数据与其对应聚类中心点距离平方和

其中

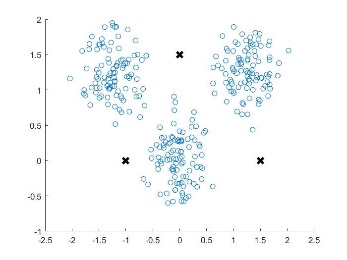
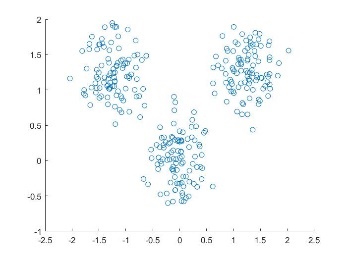
数学上已经证明这是一个NP难问题，因此*k-*Means算法采用了贪心策略，通过迭代优化来近似求解式(3.11)，具体流程如下

|  |
| --- |
| **输入**：样本集  聚类簇数*k.*  **过程**：  1：从样本集*D*中随机选择*k*个样本作为初始均指向量  2：**repeat**  3: 令  4： **fordo**  5： 计算样本与各均值向量的距离：;  6： 根据到最近的均指向量确定的簇标记：;  7： 将样本划入相应的簇：;  8： **end for**  9： **for** **do**  10： 计算新均值向量：;  11： **if then**  12： 将当前均值向量更新为  13： **else**  14： 保持当前均值向量不变  15： **end if**  16： **end for**  17：**until** 当前均值向量均未更新  **输出：**簇划分 |

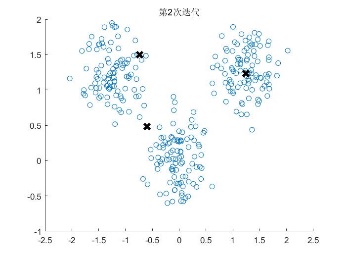
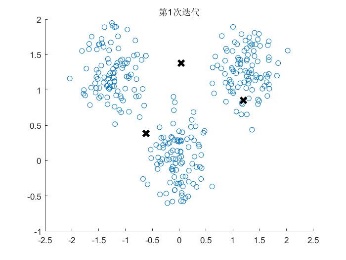
图5 *k*-Means算法

**Fig.**5 ***k*-Means algorithm**

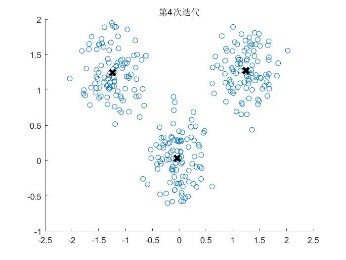
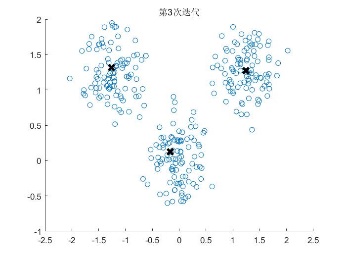
下面演示了*k-*Means算法的迭代过程：



|  |  |
| --- | --- |
| 图6 原始数据分布图 | 图7 初始化聚类中心 |
| **Fig.**6 **Original data distribution diagram** | **Fig.**7 **Initialize the clustering centers** |



|  |  |
| --- | --- |
| 图8 第一次迭代结果 | 图9 第二次迭代结果 |
| **Fig.**8 **The first iteration result** | **Fig.**9 **The second iteration result** |



|  |  |
| --- | --- |
| 图10 第一次迭代结果 | 图11 第二次迭代结果 |
| **Fig.**10 **The third iteration result** | **Fig.**11 **The fourth iteration result** |

## *k*NN

*k*NN算法是一种常用的监督学习算法，算法工作机制非常简单：给定测试集，基于某种距离度量找出与训练样本最近的*k*个样本，基于这*k*个“邻居”的信息来进行预测。通常，在分类任务中，可采用“投票法”，即选取这*k*个样本中出现最多的类别作为预测结果；在回归任务中，可采用“平均法”，即将这*k*个样本的输出平均值作为预测结果；此外，还可以根据距离远近做加权投票或加权平均，距离越近权重越大。

同*k-*Means，*k*NN算法有两个重要参数，一是参数*k*，当*k*取不同值的时候，分类结果会显著不同，另一个是距离计算方式，若采用不同的距离度量，可能找到的*k*个“近邻”会有显著差别，导致分类结果的显著不同。

*k*NN算法是一类典型的“懒惰学习”算法，即在训练阶段仅保存样本，训练时间开销为零，待收到测试样本后再进行处理；与之对应的，例如上述两个学习方法，都属于“急切学习”的范畴。

## SVM

## Naive Bayes

## EM

## BP

## CNN