**一、简介**

机器学习算法就是在没有人类干预的情况下，从数据中学习，并在经验中改善的一种方法，学习任务可能包括学习从输入映射到输出的函数，学习无标签数据的隐含结构；或者是「基于实例的学习」，通过与存储在记忆中的训练数据做比较，给一个新实例生成一个类别标签。

**二、机器学习算法的类型**

1. 监督学习:

使用有标签的训练数据去学习从输入变量（X）到输出变量（Y）的映射函数

Y = f (X)

它分为两种类型：

a. 分类：通过一个给定的输入预测一个输出，这里的输出变量以类别的形式展示。例如男女性别、疾病和健康。

b. 回归：也是通过一个给定的输入预测一个输出，这里的输出变量以实数的形式展示。例如预测降雨量、人的身高等实数值。

本文介绍的前 5 个算法就属于监督学习：线性回归、Logistic 回归、CART、朴素贝叶斯和 KNN。

集成学习也是一种监督学习方法。它意味着结合多种不同的弱学习模型来预测一个新样本。本文介绍的第 9、10 两种算法--随机森林 Bagging 和 AdaBoost 提升算法就是集成学习技术。

2. 非监督学习:

非监督学习问题仅仅处理输入变量（X），但不会处理对应的输出（也就是说，没有标签）。它使用无标签的训练数据建模数据的潜在结构。非监督学习可以分为 2 种类型：

a. 关联：就是去发觉在同一个数据集合中不同条目同时发生的概率。广泛地用于市场篮子分析。例如：如果一位顾客买了面包，那么他有 80% 的可能性购买鸡蛋。

b. 聚类：把更加相似的对象归为一类，而不是其他类别对象。

c. 降维：顾名思义，降维就是减少数据集变量，同时要保证重要信息不丢失。降维可以通过使用特征提取和特征选择方法来完成。特征选择方法会选择原始变量的一个子集。特征提取完成了从高维空间到低维空间的数据变换。例如，主成分分析（PCA）就是一个特征提取方法。

本文介绍的算法 6-8 都是非监督学习的例子：包括 Apriori 算法、K-均值聚类、主成分分析（PCA）。

3. 强化学习:

强化学习是这样一种学习方法，它允许智能体通过学习最大化奖励的行为，并基于当前状态决定下一步要采取的最佳行动。

强化学习一般通过试错学习到最佳行动。强化学习应用于机器人，机器人在碰到障碍物质之后会收到消极反馈，它通过这些消极反馈来学会避免碰撞；也用在视频游戏中，通过试错发现能够极大增长玩家回报的一系列动作。智能体可以使用这些回报来理解游戏中的最佳状态，并选择下一步的行动。

**三、监督学习**

1. 线性回归

在机器学习中，用输入变量 x 来决定输出变量 y。输入变量和输出变量之间存在一个关系。机器学习的目标就是去定量地描述这种关系。

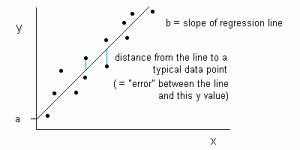
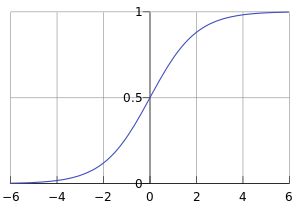


图 ：以一条直线的形式展示出来的一元线性回归：y = ax +b

线性回归的目标就是寻找参数 a 和 b 的值，一种思想是用梯度下降法对最小二乘法形式的误差函数进行优化



优点：简单，易理解

缺点：只能拟合线性数据，多重共线性

适用场景：处理数值型数据，并且预测目标数据为连续型数据

例如：根据天气中的 CO 和 NOX量 推测 O3量

收入增长，运输总量，成绩排名，搜索引擎搜索量等场景

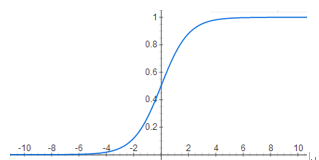
2.Logistic 回归

逻辑回归假设数据服从伯努利分布,通过极大化似然函数的方法，运用梯度

下降来求解参数，来达到将数据二分类的目的。

在LR中，将线性回归的结果通过sigmoid函数映射到0到1之间，映射的结果刚好可以看做是数据样本点属于某一类的概率，如果结果越接近0或者1，说明分类结果的可信度越高。这样做不仅应用了线性回归的优势来完成分类任务，而且分类的结果是0~1之间的概率，可以据此对数据分类的结果进行打分。输出是默认类别的概率，值域是 [0,1]，然后使用一个阈值强制地让输出结果变成一个二元分类问题。

  sigmoid函数形式为 = ， 分布如下图所示：



LR推导与证明：

线性回归

Sigmoid函数作用： ，值域是(0,1),

可以将 看作是 类别为1 的概率。二项LR模型由条件概率分布P(Y|X)表示，随机变量X取值为实数，随机变量Y取值为1或0。模型的条件概率分布：

也就是说，LR将输入实例x的线性回归的加权结果通过sigmoid函数映射到0~1之间，加权结果的值越接近正无穷，概率值就越接近1，反之则越接近0。注上面公式的y是回归预测值，下面公式的y是Y的取值（0或1）：

似然函数： =

取对数 ：

对数损失函数：L(w,b) =

使似然函数最大，就是使损失函数最小，损失函数是凸函数，利用梯度下降法求最优参数，sigmoid函数很方便的一点还在于

权值w的更新过程中，推导如下：

基于此，迭代公式为:

这个式子的更新速度只和，相关,和sigmoid函数本身的梯度是无关的。这样更新的速度是可以自始至终都比较的稳定。平方损失函数，梯度更新的速度和sigmod函数本身的梯度是很相关的,sigmod函数在它在定义域内的梯度都不大于0.25,这样训练会非常的慢,而且最小平方损失函数是非凸函数，梯度下降可能只求出局部最优解。

优点：解释性强，输出值自然地落在0到1之间，并且有概率意义，适合需要得到一个分类概率的场景；计算代价不高，LR在时间和内存需求上相当高效，可以应用于分布式数据，并且还有在线算法实现，用较少的资源处理大型数据；对数据中小噪声的鲁棒性很好，并且不会受到轻微的多重共线性的特别影响；拟合出来的参数就代表了每一个特征(feature)对结果的影响，是一个理解数据的好工具。

缺点：容易欠拟合，因为它本质上是一个线性的分类器，所以处理不好特征之间相关的情况（严重的多重共线性则可以使用逻辑回归结合L2正则化来解决，但是若要得到一个简约模型，L2正则化并不是最好的选择，因为它建立的模型涵盖了全部的特征。可以利用因子分析或者变量聚类分析等手段来选择代表性的自变量，以减少候选变量之间的相关性）；不能很好地处理大量多类特征；对缺失值敏感；预测结果呈“S”型，因此从log(odds)向概率转化的过程是非线性的，在两端随着​log(odds)值的变化，概率变化很小，而中间概率的变化很大，很敏感。 导致很多区间的变量变化对目标概率的影响没有区分度，无法确定阀值；很难处理数据不平衡的问题。

应用场景：线性可分问题。

LR是解决工业规模问题最流行的算法，在工业应用上，如果需要分类的数据拥有很多有意义的特征，每个特征都对最后的分类结果有或多或少的影响，那么最简单最有效的办法就是将这些特征线性加权，一起参与到决策过程中；

预测是否发生、发生的概率（流失客户预测）；

影响因素分析（找出影响结果的主要因素）考察某单一因素是否为影响某一事件发生与否的因素

准确率，召回率，混淆(错判)矩阵 等评价分类效果。

参数解释（对变量的评价）

发生比(odds)： ODDS=事件发生概率/事件不发生的概率=P/(1-P)

发生比率（odds ratio）：odds ratio=oddsB/oddsA (组B相对于组A更容易发生的比率）

注：odds ratio大于1或者小于1都有意义，代表自变量的两个分组有差异性，对因变量的发生概率有作用。若等于1的话，该组变量对事件发生概率没有任何作用。

逻辑回归不是对一个独立的因变量本身进行线性拟合，而是对odds的对数进行线性拟合。正向变动一个单位，发生比变动个单位，>0,发生比上升。逻辑回归实际上就是以线性回归的形式去逼近事件发生优势比的对数

与SVM比较：

线性回归做分类，因为考虑了所有样本点到分类决策面的距离，所以在两类数据分布不均匀的时候将导致误差非常大。LR和SVM克服了这个缺点，其中LR将所有数据采用sigmod函数进行了非线性映射，使得远离分类决策面的数据作用减弱；SVM直接去掉了远离分类决策面的数据，只考虑支持向量的影响。但是对于这两种算法来说，在线性分类情况下，如果异常点较多无法剔除的话，LR中每个样本都是有贡献的，最大似然后会自动压制异常的贡献；SVM+软间隔对异常比较敏感，因为其训练只需要支持向量，有效样本本来就不高，一旦被干扰，预测结果难以预料。

3. 分类和回归树CART

参考https://www.cnblogs.com/pinard/p/6050306.html

决策树的构建是基于样本概率和纯度进行构建操作的，判断数据集是否“纯”可以通过三个公式进行判断，分别是Gini系数、熵(Entroy)、错误率。

CART采用Gini系数来度量分裂时的不纯度，比信息熵的计算速度更快一些。基尼系数越小，则不纯度越低，特征越好。这和信息增益(比)是相反的。sk-learn决策树采用的是优化的CART。在计算机中二叉树模型会比多叉树运算效率高。CART采用二叉树模型，ID算法采用多叉树模型。

假设有n个类别，第k个类别的概率为,则基尼系数的表达式为：

样本D,第k个类别的数量为：

如果根据特征A的某个值a,把D分成D1和D2两部分，则在特征A的条件下，D的基尼系数表达式为

Gini系数越大，样本集合的不确定性程度越高。分类学习过程的本质是样本不确定性程度的减少（即熵减过程），故应选择最小Gini指数的特征分裂。

CART算法步骤：

算法输入是训练集D，基尼系数的阈值，样本个数阈值。输出是决策树T。

　　算法从根节点开始，用训练集递归的建立CART树。

　　1.对于当前节点的数据集为D，如果样本个数小于阈值或者没有特征，则返回决策子树，当前节点停止递归。

　　2.计算样本集D的基尼系数，如果基尼系数小于阈值，则返回决策树子树，当前节点停止递归。

3.计算当前节点现有各个特征的各个特征值对数据集D的基尼系数　　　　 4.选择基尼系数最小的特征A和对应的特征值a，把数据划分为两部分D1

和D2，同时建立当前节点的左右节点D1和D2

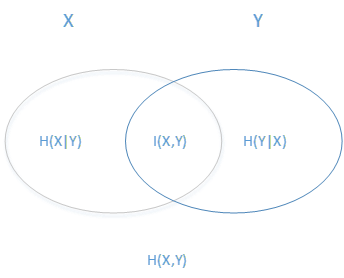
5.对左右的子节点递归的调用1-4步，生成决策树。

6.剪枝

7.预测。预测时，假如测试集里的样本A落到了某个叶子节点，而节点里有多个训练样本，A的类别预测采用的是这个叶子节点里概率最大的类别。

熵度量了事物的不确定性，越不确定的事物，它的熵就越大。具体的，随机变量X的熵的表达式如下：

决策树算法用的是信息增益：



在相同条件下，取值比较多的特征比取值少的特征信息增益大。比如一个变量有2个值，各为1/2，另一个变量为3个值，各为1/3，其实他们都是完全不确定的变量，但是取3个值的比取2个值的信息增益大。对于信息增益作为标准容易偏向于取值较多的特征的问题。引入一个信息增益比的变量——信息增益和特征熵的比值。D为样本特征输出的集合，A为样本特征,表达式如下：

特征熵:

特征数越多的特征对应的特征熵越大，它作为分母，可以校正信息增益容易偏向于取值较多的特征的问题，这就是决策树选择划分特征的标准。熵模型大量的对数运算明显比基尼系数的运算多，基尼系数和熵之半的曲线非常接近，仅仅在45度角附近误差稍大。因此，基尼系数可以做为熵模型的一个近似替代。

优点：可分类，可回归（连续的特征离散化）；简单直观，不需要预处理，提前归一化，处理缺失值。

缺点：不支持在线学习，新样本到来后，决策树需要全部重建；容易出现过拟合，可通过剪枝，设置节点最少样本数量和限制决策树深度来改进；如果某些特征的样本比例过大，生成决策树容易偏向于这些特征，需要通过调节样本权重来改善。所谓多值偏向是指决策树算法在选择拆分特征时，倾向于优先选择取值个数较多的特征，多值偏向所带来的问题是把特征在分类中的重要性跟特征取值个数的多少关联起来。

应用场景：分类与回归，比如预测天气情况

分析对某种响应可能性影响最大的因素（判断具有什么特征的客户流失概率更高）

为其他模型筛选变量（决策树找到的变量是对目标变量影响很大的变量，所以可以作为筛选变量的手段）

4. 朴素贝叶斯

事件A1，A2，…，An构成一个完备事件组且都有正概率

全概率公式： 原因推结果

**贝叶斯公式： 结果推原因**

假设某个体有n项特征F1、F2、…、Fn，有m个类别C1、C2、…、Cm。贝叶斯分类器就是计算出概率最大的那个(朴素贝叶斯假设各特征独立)，即使下面公式达到最大值的类别

根据先验概率，和求后验概率。虽然“所有特征彼此独立”这个假设，在现实中不太可能成立，但是它可以大大简化计算，特征不是很相关时对分类结果的准确性影响不大。

在估计先验概率与条件概率时，有可能出现为0的情况，则计算得到的后验概率亦为0，从而影响分类的效果。因此，需要在估计时做拉普拉斯平滑（Laplace Smoothing），也就是参数为1时的**贝叶斯估计**。初始时将所有单词的计数设置为1，防止单词个数为0的情况出现。它的思想非常简单，就是对先验概率的分子（划分的计数）加1，分母加上类别数；对条件概率分子加1，分母加上对应特征的可能取值数量。这样在解决零概率问题的同时，也保证了概率和依然为1。比如：假设在文本分类中，有3个类，C1、C2、C3，在指定的训练样本中，某个词语F1，在各个类中观测计数分别为=0，990，10，即概率为P(F1/C1)=0，P(F1/C2)=0.99，P(F1/C3)=0.01，对这三个量使用拉普拉斯平滑的计算方法：1/1003 = 0.001，991/1003=0.988，11/1003=0.011

朴素贝叶斯常用的三个模型有：

高斯模型：处理特征是连续型变量的情况

多项式模型：最常见，要求特征是离散数据

伯努利模型：要求特征是离散的，且为布尔类型，即true和false，或者1，0

#### 多项式模型：特征：单词，值：k类单词出现频次

#### 在多项分布朴素贝叶斯模型中，特征向量 X 的特征 通常为 离散型变量，并且假定所有特征的取值是符合多项分布的，可用于文本分类。

#### 在多项式模型中，设某文档d=(t1,t2,…,tk)，tk是该文档中出现过的单词，允许重复，则类先验概率P(c)= 类c下词总数/整个训练样本的单词总数。类条件概率\*\*P(tk|c)=(类c下单词tk在各个文档中出现过的次数之和+1)/(类c下单词总数+|V|)。V是训练样本的单词表（即抽取单词，单词出现多次，只算一个），|V|则表示训练样本包含多少种单词。P(tk|c)可以看作是单词tk在证明d属于类c上提供了多大的证据，而P(c)则可以认为是类别c在整体上占多大比例(有多大可能性)。

#### 优点：算法简单,快速, 具有较小的出错率；有稳定的分类效率，对小规模的数据表现很好，能个处理多分类任务，适合增量式训练，尤其是数据量超出内存时，可以一批批的去增量训练；缺失数据不太敏感，

缺点： 理论上，朴素贝叶斯模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为朴素贝叶斯模型假设属性之间相互独立，这个假设在实际应用中往往是不成立的，在属性个数比较多或者属性之间相关性较大时，分类效果不好。而在属性相关性较小时，朴素贝叶斯性能最为良好。对于这一点，有半朴素贝叶斯之类的算法通过考虑部分关联性适度改进；

要知道先验概率，且先验概率很多时候取决于假设，假设的模型有很多种，因此可能会由于假设的先验模型的原因导致预测效果不佳。

通过先验和数据来决定后验的概率从而决定分类，所以分类决策存在一定的错误率。

对输入数据的表达形式很敏感。

应用场景：属于监督学习的生成模型，实现简单，没有迭代，学习效率高，在大样本量下会有较好的表现。但因为假设太强——假设特征条件独立，在输入向量的特征条件有关联的场景下并不适用。主要应用于病人分类，拼写检查，电子邮件过滤以及文本分类等。

5.KNN

KNN算法的核心思想非常简单：在训练集中选取离输入的数据点最近的k个邻居，根据这个k个邻居中最大表决规则预测。

三要素：距离度量，k的大小和分类规则。

当训练数据集和三要素确定后，相当于将特征空间划分成一些子空间，对于每个训练实例xi，距离该点比距离其它点更近的所有点组成了一个区域，每个区域的类别由决策规则确定且唯一，从而将整个区域划分。对于任何一个测试点，找到其所属的子空间，其类别即为该子空间的类别。

距离度量：常用闵可夫斯基距离(Minkowski Distance)

p>=1,p=2是欧氏距离，p=1是曼哈顿距离

二、K的选择：k的选择会对算法的结果产生重大影响。

如果k值较小，就相当于用较小邻域中的训练实例进行预测，极端情况下k=1，测试实例只和最接近的一个样本有关，训练误差很小(0)，但是如果这个样本恰好是噪声，预测就会出错，测试误差很大。也就是所，当k值较小的，会产生过拟合的现象。

如果k值较大，就相当于用很大邻域中的训练实例进行预测，极端情况是k=n，测试实例的结果是训练数据集中实例最多的类，这样会产生欠拟合。

三、**分类规则：**

决策规则通常是多数表决，即由测试样本的k个临近样本的多数类决定测试样本的类别。给定测试样本x,其最邻近的k个训练实例构成集合，分类损失函数为0-1损失,优化策略是经验风险最小，多数表决的风险最小。

优点：简单；重新训练的代价较低（类别体系的变化和训练集的变化，在Web环境和电子商务应用中是很常见的）；由于KNN方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，KNN方法较其他方法更为适合；该算法比较适用于样本容量比较大的类域的自动分类，而那些样本容量较小的类域采用这种算法比较容易产生误分。

缺点：懒散学习方法（lazy learning,基本上不学习），一些积极学习的算法要快很多；类别评分不是规格化的（不像概率评分）输出的可解释性不强；当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的K个邻居中大容量类的样本占多数。该算法只计算“最近的”邻居样本，某一类的样本数量很大，那么或者这类样本并不接近目标样本，或者这类样本很靠近目标样本。无论怎样，数量并不能影响运行结果。可以采用权值的方法（和该样本距离小的邻居权值大）来改进k邻域的样本点对预测结果的贡献度是相等的；但距离更近的样本点应有更大的相似度，其贡献度应比距离更远的样本点大。可以加上权值wi=1/∥xi−x∥进行修正

#### 应用场景：分类算法，适用于对样本容量比较大的类域进行自动分类。

#### 约会网站的数据分类，改进约会网站的配对效果；手写数字识别

四、非监督学习算法:

6.Apriori 算法

Apriori算法是一种挖掘关联规则的算法，用于挖掘其内涵的、未知的却又实际存在的数据关系，其核心是基于两阶段频集思想的递推算法。先寻找频繁项集再由频繁项集找关联规则。

令I={i1,i2,i3……id}是购物篮数据中所有项的集合，而T={t1,t2,t3….tN}是所有事务的集合，在超市购物分析中，一个顾客的一次购买记录（比如啤酒和尿布值为1，其他商品值为0）就是一个事务，每个事务ti包含的项集都是I的子集。在关联分析中，包含0个或多个项的集合称为项集。如果一个项集包含K个项，则称它为K-项集。空集是指不包含任何项的项集。用一个简单的例子说明。表6-1是顾客购买记录的数据库D，包含6个事务。项集I={网球拍,网球,运动鞋,羽毛球}。



项集 → 项集。项集X出现的事务次数（亦称为support count）定义为：

其中，表示某个事务（TID），T表示事务的集合。支持度(support)如下：

支持度刻画了项集X∪Y的出现频次。置信度（confidence）定义如下：

置信度可理解为条件概率P(Y|X)，度量在已知事务中包含了X时包含Y的概率。对于靠谱的关联规则，其支持度与置信度均应大于设定的阈值。那么，关联分析问题即等价于：对给定的支持度阈值min\_sup、置信度阈min\_conf，找出所有的满足下列条件的关联规则：

支持度>=min\_sup；置信度>=min\_conf

把支持度大于阈值的项集称为频繁项集（frequent itemset）。因此，关联规则分析可分为下列两个步骤：

一、生成频繁项集F=X∪Y

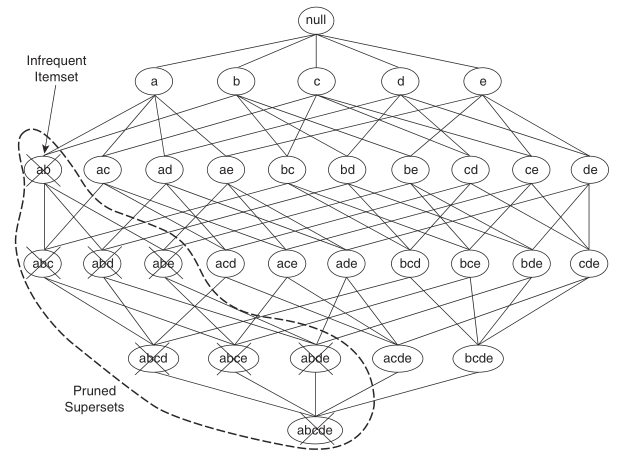
二、在频繁项集F中，找出所有置信度大于最小置信度的关联规则X⟶Y。

如果要穷举所有的关联规则，时间复杂度将达到指数级，因此需要找出时间复杂度较低的算法，比如Apriori算法用于做快速的关联分析。

先验定理：

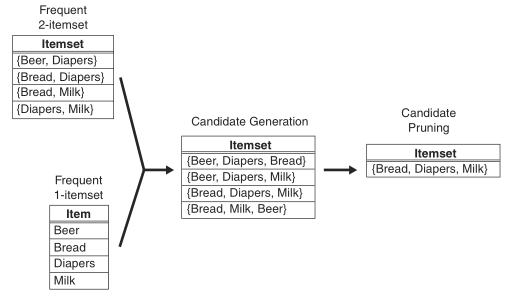
定理一：如果一个项集是频繁的，那么其所有的子集(subsets)也一定是频繁的。因为某项集的子集的支持度一定不小于该项集。

定理二：同一理，如果一个项集是非频繁的，那么其所有的超集（supersets）也一定是非频繁的，根据定理二，可以进行如下的剪枝：

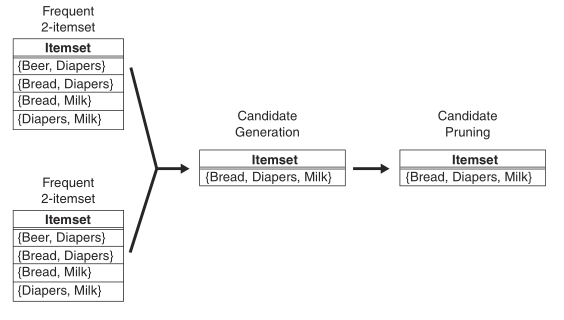


对于大小为k−1的频繁项集，如何计算大小为k的频繁项集呢？Apriori算法给出了两种策略

1. 方法。之所以没有选择与所有1项集生成，是为了满足定理2。下图给出由频繁项集与生成候选项集：

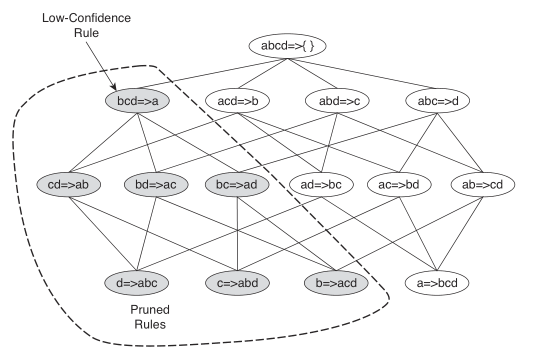


二、 选择前k−2项均相同的进行合并，生成。当然，的所有都是有序排列的。之所以要求前k−2项均相同，是为了确保的k−2项都是频繁的。下图给出由两个频繁项集F2生成候选项集C3：



关联规则是由频繁项集生成的，即对于，找出项集，使得规则

− ⟶ 的置信度大于置信度阈值。同样地，根据置信度定义得到定理3：如果规则X ⟶ Y−X 不满足置信度阈值，则对于X的子集X′，规则 X′⟶Y−X′也不满足置信度阈值。根据定理3，可对规则树进行如下剪枝：



优点：使用先验性质，大大提高了频繁项集逐层产生的效率；简单易理解；数据集要求低。它可以产生清晰有用的结果它支持间接数据挖掘可以处理变长的数据它的计算的消耗量是可以预见的当问题变大时，

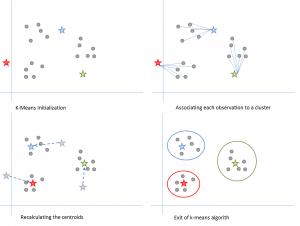
缺点：在每一步产生侯选项目集时循环产生的组合过多,没有排除 不应该参与组合的元素；每次计算项集的支持度时,都对数据库中的全部记录进行了一遍扫描比较,需要很大的I/O 负载计算量增长得厉害容易忽略稀有的数据

应用场景：只要一个客户在同一个时间里买了多样东西，或者在一段时间了做了好几样事情就可能是一个潜在的应用。例如：用信用卡购物，如汽车租金和旅馆费，可以看他下一个要买的东西； 电话公司提供的多项服务，以研究捆绑销售的问题；银行提供的多项服务，来分析客户可能需要那些服务；不寻常的多项保险申请可能是欺诈行为；如果一个顾客购买了商品 X 之后又购买了商品 Y，那么这个关联规则就可以写为：X -> Y

改进：

7.K-均值聚类算法

k-means与kNN虽然都是以k打头，但却是两类算法——kNN为监督学习中的分类算法，而k-means则是非监督学习中的聚类算法；二者均利用近邻信息。K-均值是一个迭代算法。它计算出 k 个聚类的中心点，并给某个类的聚类分配一个与其中心点距离最近的数据点。



步骤：

选取k个初始质心（作为初始cluster）；

repeat：

对每个样本点，计算得到距其最近的质心，将其类别标为该质心所对应的cluster；

重新计算k个cluser对应的质心；

until 质心不再发生变化

对于欧式空间的样本数据，以平方误差和（sum of the squared error, SSE)作为聚类的目标函数，同时也可以衡量不同聚类结果好坏的指标。最优的聚类结果应使得SSE达到最小值。

优点：算法简单，容易实现 ;对处理[大数据](http://lib.csdn.net/base/spark)集，该算法是相对可伸缩的和高效率的，因为它的复杂度大约是O(nkt)，其中n是所有对象的数目，k是簇的数目,t是迭代的次数。通常k<<n。

缺点：适合数值型数据，k-means通常是局部最优的，大规模数据集收敛慢，容易受到初始质心的影响；同时，k值的选取也会直接影响聚类结果，最优聚类的k值应与样本数据本身的结构信息相吻合，而这种结构信息是很难去掌握，因此选取最优k值是非常困难的。

应用场景：算法尝试找出使平方误差函数值最小的k个划分。当簇是密集的、球状或团状的，且簇与簇之间区别明显时，聚类效果较好。除了一般的聚类场景（例如对用户进行分群组、对微博文本进行分群组等）以外，可以用KMeans实现单变量的离散化，因为一般的等频和等距的离散化方法往往会忽略变量中潜在的分布特征，而基于聚类的离散化可以一定程度地保留变量的分布特征。

优化：在基本k-means的基础上发展而来二分 ，其主要思想：一个大cluster进行分裂后可以得到两个小的cluster；为了得到k个cluster，可进行k-1次分裂。算法流程如下：

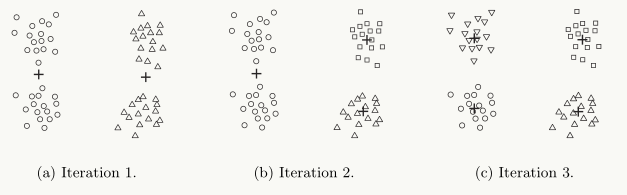
初始只有一个cluster包含所有样本点

repeat:

从待分裂的clusters中选择一个进行二元分裂，所选的cluster应使得SSE最小；

until 有k个cluster

上述算法流程中，为从待分裂的clusters中求得局部最优解，可以采取暴力方法：依次对每个待分裂的cluster进行二元分裂（bisect）以求得最优分裂。二分k-means算法对初始质心的选择不太敏感，因为初始时只选择一个质心。二分k-means算法聚类过程如图：



8. 主成分分析（PCA）

PCA的思想是将n维特征映射到k维空间上k<n，这k维特征是全新的正交特征，是重新构造出来的k维特征，而不是简单地从n维特征中去除其余n−k维特征。把多指标转化为少数几个[综合指标](https://baike.baidu.com/item/%E7%BB%BC%E5%90%88%E6%8C%87%E6%A0%87/2849120)（即主成分），其中每个主成分都能够反映原始变量的大部分信息，且所含信息互不重复。这种方法在引进多方面变量的同时将复杂因素归结为几个主成分，使问题简单化，同时得到的结果更加科学有效的数据信息。在实际问题研究中，为了全面、系统地分析问题，必须考虑众多影响因素。因为每个变量都在不同程度上反映了所研究问题的某些信息，并且指标之间彼此有一定的相关性，因而所得的统计数据反映的信息在一定程度上有重叠。主成分分析（PCA）通过减少变量的数目来使数据变得更加易于探索和可视化。这通过将数据中拥有最大方差的数据抽取到一个被称为「主成分」的新坐标系中。每一个成分都是原始变量的一个新的线性组合，且是两两统计独立的。统计独立意味着这些成分的相关系数是 0。

第一主成分捕获的是数据中最大方差的数据。第二主成分捕获的是剩下的数据中方差最大但是与第一主成分相互独立的数据。相似地，后续的主成分（例如 PC3、PC4）都是剩下的数据中方差最大的但是与之前的主成分保持独立的数据。

最大方差理论：在信号处理中认为信号具有较大的方差，噪声有较小的方差，信噪比就是信号与噪声的方差比，越大越好。因此我们认为，最好的k维特征是将n维样本点变换为k维后，每一维上的样本方差都尽可能的大。方差越大，包含的原始信息越多。

优点：无监督学习，完全无参数限制的，最后的结果只与数据相关，与用户是独立的；根据需要取前面最重要的主成分，将后面的维数省去，可以达到降维从而简化模型或是对数据进行压缩的效果。同时最大程度的保持了原有数据的信息；各主成分之间正交，可消除原始数据成分间的相互影响。

缺点：如果用户对观测对象有一定的先验知识，掌握了数据的一些特征，却无法通过参数化等方法对处理过程进行干预，可能会得不到预期的效果，效率也不高；贡献率小的主成分往往可能含有对样本差异的重要信息；在非高斯分布的情况下，PCA方法得出的主元可能并不是最优的，此时在寻找主元时不能将方差作为衡量重要性的标准。

应用场景：降维，比如人脸识别中提取特征脸。对于一张n\*n的人脸图片，因为人脸的结构有极大的相似性（特别是同一个人的人脸图像），则使用PCA方法就可以很容易的提取出人脸的内在结构，也及时所谓“模式”，如果有新的图像需要与原有图像比较，就可以在变换后的主元维度上进行比较，则可衡量新图与原有数据集的相似度如何。对这样的一组人脸图像进行处理，提取其中最重要的主元，即可大致描述人脸的结构信息，称作“特征脸”。这就是人脸识别中的重要方法“特征脸方法”的理论根据。

主成分分析与因子分析区别：

PCA基本原理：线性变换的思想，在损失很少信息的前提下把多个指标转化为几个不相关的综合指标（主成分),即每个主成分都是原始变量的线性组合，从而达到简化系统结构，抓住问题实质的目的。

FA基本原理：利用降维的思想，由研究原始变量相关矩阵内部的依赖关系出发，把一些具有错综复杂关系的变量表示成少数的公共因子和仅对某一个变量有作用的特殊因子线性组合而成。就是要从数据中提取对变量起解释作用的少数公共因子（因子分析是主成分的推广，相对于主成分分析，更倾向于描述原始变量之间的相关关系）

主成分分析：重点在于解释个变量的总方差；  
因子分析：则把重点放在解释各变量之间的协方差。

五、集成学习技术:

集成学习的理论基础基于PAC（probably approximately correct）的可学习性理论 ，PAC 定义了学习算法的强弱：

弱学习算法：识别错误率小于1/2(即准确率仅比随机猜测略高的算法)

强学习算法：识别准确率很高并能在多项式时间内完成的算法

根据这两个概念，后来产生了一个重要的结论：

强可学习与弱可学习是等价的，即：一个概念是强可学习的充要条件是这个概念是弱可学习的。

据此，为得到一个优秀的强学习模型，可以将多个简单的弱学习模型“提升”。

集成学习面临两个主要问题：   
1.如何调整输入训练数据的概率分布及权值   
2.如何将多个弱分类器组合成一个强分类器

针对上述问题，目前主流方法有三种：   
1.Boosting(提升)：包括Adaboosting，提升树（代表是GBDT）, XGBoost 2.Bagging(装袋)： 典型的是随机森林   
3.Stacking(堆叠)算法

**经典的样本估计方法Bootstrapping：**

**Bootstrapping**一种**有放回的抽样方法**。它是非参数统计中一种重要的通过估计统计量方差进而进行区间估计的统计方法，遵从“在不知道数据总体分布时，对总体分布的最好的猜测便是由数据提供的分布”原则。自助法的要点是：   
 1.假定观察值就是数据总体   
 2.由这一假定的总体抽取样本，

具体步骤如下：

1.采用重抽样方式从原始样本中抽取一定数量（容量m）的数据，重复抽样m次，得到一个自助样本。

2.根据得到的自助样本计算特定的统计量。

3.重复上述N次，得到N个自助统计量。

4.根据上述N个自助统计量估计出真实值。

bootstrap实质上是一种再抽样过程，相对于其他方法，在小样本时也具有较好效果。

Boosting方法：

Boosting方法是典型的bootstrapping思想的应用，其特点是：每一次迭代时训练集的选择与前面各轮的学习结果有关，而且每次是通过更新各个样本权重的方式来改变数据分布。总结起来如下：

1.分步学习每个弱分类器，最终的强分类器由分步产生的分类器组合而成

2.根据每步学习到的分类器去改变各个样本的权重（被错分的样本权重加大，反之减小)

Bagging方法：

同一个学习算法在来自同一分布的多个不同的训练数据集上训练得到的模型偏差可能较大，即模型的方差（variance）较大，为了解决这个问题，可以综合多个模型的输出结果，对于回归问题可以取平均值，对于分类问题可以采取多数投票的方法。这就是Bagging的核心思想。要想综合N个弱学习模型的结果，需要采样N个训练数据集，在实际应用中获取N个训练数据集往往不现实，BootStrap 采样提供了一种有效的解决方法。有放回的随机采样，同cross-validation一样，是一种resample技术。

Bagging与Boosting的区别：

1. bagging的训练集是随机的，各训练集是独立的；而boosting训练集的选择不是独立的，每一次选择的训练集都依赖于上一次学习的结果
2. bagging的每个预测函数都没有权重；而boosting根据每一次训练的训练误差得到该次预测函数的权重
3. bagging的各个预测函数可以并行生成；而boosting只能顺序生成(用强力工具是可以并行的)。（对于神经网络这样极为耗时的学习方法，bagging可通过并行训练节省大量时间开销）
4. 从“偏差-方差分解”的角度看，Bagging关注于降低variance，而Boosting则是降低bias

关于偏差与方差：

偏差-方差分解”（bias variance decomposition）是用来解释机器学习算法的泛化能力的一种重要工具。对于同一个算法，在不同训练集上学得结果可能不同。对于训练集D={(,),(,),⋯,(,)}，由于噪音，样本x的真实类别为（在训练集中的类别为y），则噪声为:

学习算法的期望预测为:

例如用不同的数据集得到

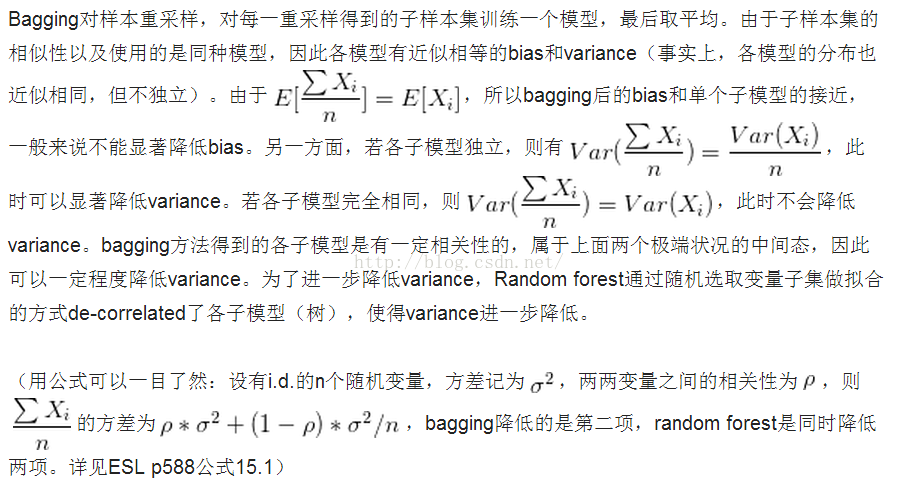
使用样本数相同的不同训练集所产生的方法:

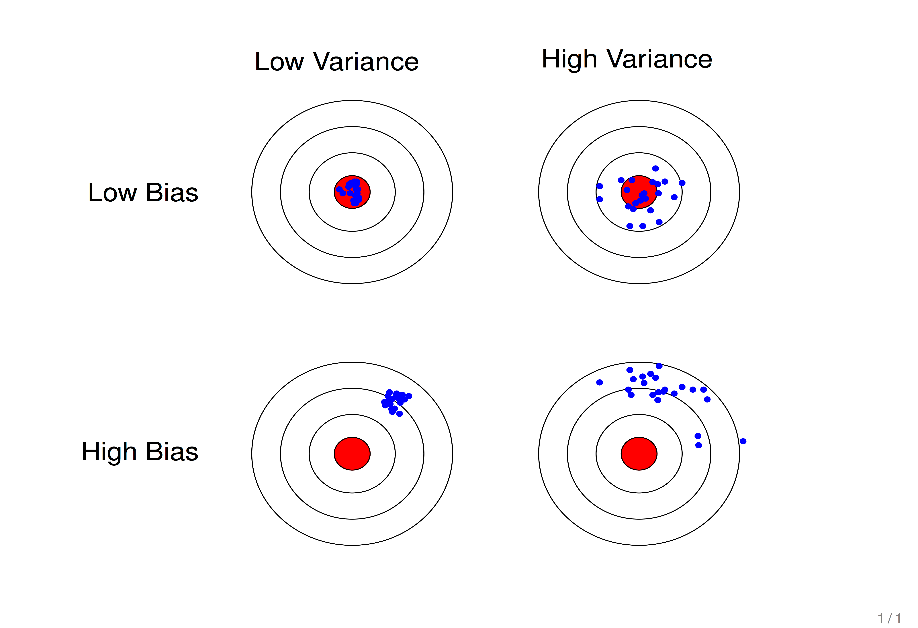
期望输入与真实类别的差别称为bias，则:

为便于讨论，假定噪声的期望为0，即,（实际y和通常一样？）通过多项式展开，可对算法的期望泛化误差进行分解:

也就是说，误差可以分解为3个部分：bias、variance、noise。bias度量了算法本身的拟合能力，刻画模型的**准确性**；variance度量了数据扰动所造成的影响，刻画模型的**稳定性**。为了取得较好的泛化能力，则需要充分拟合数据（bias小），并受数据扰动的影响小（variance小）。但是，bias与variance往往是不可兼得的

* 当训练不足时，拟合能力不够强，数据扰动不足以产生较大的影响，此时bias主导了泛化错误率；
* 随着训练加深时，拟合能力随之加强，数据扰动渐渐被学习到， variance主导了泛化错误率。





Bagging和Boosting都可以视为比较传统的集成学习思路。 现在常用的Random Forest，GBDT（迭代决策树），GBRank其实都是更加精细化，效果更好的方法。

9.随机森林 Bagging + 决策树

“**随机**”和“**森林**”，一个使它具有抗过拟合能力，一个使它更加精准。

在生成每棵树的时候，每个树选取的特征都仅仅是随机选出的少数特征，一般默认取特征总数m的开方。而一般的CART树则是会选取全部的特征进行建模。因此，不但样本是随机的，也保证了**特征随机性**。由于随机性，对于降低模型的方差很有作用，故随机森林一般不需要额外做剪枝，即可以取得较好的泛化能力和抗过拟合能力（Low Variance）。当然对于训练集的拟合程度就会差一些，也就是模型的偏倚会大一些（High Bias），**仅仅是相对的**。

抗过拟合

首先，正如Bagging介绍中提到的，每个树选取使用的特征时，都是从全部m个特征中随机产生的，本身已经降低了过拟合的风险和趋势。模型不会被特定的特征值或者特征组合所决定，随机性的增加，将控制模型的拟合能力不会无限提高。第二，与决策树不同，RF对决策树的建立做了改进。对于普通的决策树，我们会在节点上所有的m个样本特征中选择一个最优的特征来做决策树的左右子树划分。但是RF的每个树，其实选用的特征是一部分，在这些少量特征中，选择一个最优的特征来做决策树的左右子树划分，将随机性的效果扩大，进一步增强了模型的泛化能力。假设每棵树选取msub个特征，msub越小，此时模型对于训练集的拟合程度会变差，偏差增加，但是会泛化能力更强，模型方差减小。msub越大则相反。在实际使用中，一般会将msub的取值作为一个参数，通过开启oob验证或使用交叉验证，不断调整参数以获取一个合适的msub的值。

Bagging的算法过程如下：

1.从原始样本集中使用Bootstraping方法随机抽取n个训练样本，共进行k轮抽取，得到k个训练集。（k个训练集之间相互独立，元素可以有重复）

2.对于k个训练集，我们训练k个模型（这k个模型可以根据具体问题而定，比如决策树，knn等）

3.对于分类问题：由投票表决产生分类结果；对于回归问题：由k个模型预测结果的均值作为最后预测结果。（所有模型的重要性相同）

优点：集成算法，本身精度比大多数单个算法要好；在测试集上表现良好，由于两个随机性的引入，使得随机森林不容易陷入过拟合（样本随机，特征随机），在工业上，由于两个随机性的引入，使得随机森林具有一定的抗噪声能力，对比其他算法具有一定优势；能处理很高维度的数据，并且不用做特征选择；既能处理离散型数据，也能处理连续型数据，数据集无需规范化 ；训练速度快，可以得到变量重要性排序 ；容易实现并行化

缺点：在某些噪音较大的分类或回归问题上会过拟

对于有不同取值的属性的数据，取值划分较多的属性会对随机森林产生更大的影响，所以随机森林在这种数据上产出的属性权值是不可信的。

应用场景：因为不需要很多参数调整就可以达到不错的效果，基本上不知道用什么方法的时候都可以先试一下随机森林。

10． Adaboost( adaptive boosting）

Adaboost是一种加法模型，损失函数是指数函数算法过程简述为：前一个分类器改变权重w后，成为后一个新的分类器。如果一个训练样本在前一个分类器中被误分，那么它的权重会被加大，相反，被正确分类的样本权重会降低。即：AdaBoost算法通过给已有模型预测错误的样本更高的权重，使得之前的分类模型出错的训练样本在后续受到更多关注的方式来弥补已有模型的不足。通过每一次迭代改变训练数据的权值分布，使得数据在每一个基本分类器的学习中起到不同作用，从而使不同数据发挥各自不同的作用，因此不易发生过拟合。相关的训练误差分析表明，每一次迭代后构建的分类器，其分类误差率随着迭代次数增加而稳定下降。

Adaboost优点：

高精度的分类器；可以用各种方法构建子分类器，adaboost算法提供的是框架

当使用简单分类器时，计算出的结果是可以理解的，而且弱分类器构造极其简单；

简单，不用做特征筛选；不用担心overfitting！

Adaboost应用场景：

用于二分类或多分类的应用场景