**1.PageRank算法**

算法是做什么的？PageRank是为了决定一些对象和同网络中的其他对象之间的相对重要程度而设计的连接分析算法(link analysis algorithm)。

那么什么是连接分析算法呢？它是一类针对网络的分析算法，探寻对象间的关系（也可成为连接）。

举个例子：最流行的 PageRank 算法是 Google 的搜索引擎。尽管他们的搜索引擎不止是依靠它，但 PageRank依然是 Google 用来测算网页重要度的手段之一。

解释一下：

万维网上的网页都是互相链接的。如果 Rayli.net 链接到了 CNN 上的一个网页，CNN 网页就增加一个投票，表示 rayli.net 和 CNN 网页是关联的。

这还没有结束：

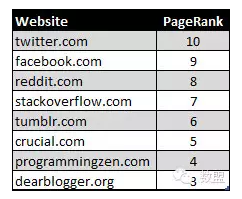
反过来，来自rayli.net 网页的投票重要性也要根据 rayli.net 网的重要性和关联性来权衡。换句话说，任何给 rayli.net 投票的网页也能提升 rayli.net 网页的关联性。

基本概括一下：

投票和关联性就是 PageRank 的概念。rayli.net 给CNN 投票增加了 CNN 的 Pagerank，rayli.net 的 PageRank级别同时也影响着它为 CNN 投票多大程度影响了CNN 的 PageRank。

那么 PageRank 的0，1，2，3级别是什么意思？ 尽管 Google 并没有揭露PageRank 的精确含义，我们还是能了解它的大概意思。

我们能通过下面这些网站的PageRank得到些答案：



看到了么？

这排名有点像一个网页流行度的竞争。我们的头脑中都有了一些这些网站的流行度和关联度的信息。

PageRank只是一个特别讲究的方式来定义了这些而已。

PageRank还有什么其他应用呢？ PageRank是专门为了万维网设计的。

可以考虑一下，以核心功能的角度看，PageRank算法真的只是一个处理链接分析极度有效率的方法。处理的被链接的对象不止只是针对网页。

下面是 PageRank3个创新的应用：

1. 芝加哥大学的Dr Stefano Allesina，将 PageRank应用到了生态学中，测定哪个物种对可持续的生态系统至关重要。
2. Twitter 研究出了一种叫 WTF（Who-to-Follow）算法，这是一种个性化的 PageRank推荐关注人的引擎。
3. 香港理工大学的 Bin Jiang 使用一种变形的PageRank来预测基于伦敦地形指标的行人移动速率。

这算法是监督的还是非监督的？PageRank常用来发现一个网页的重要度关联度，通常被认为是一种非监督学习算法。

为什么使用PageRank？可以说，PageRank的主要卖点是：由于得到新相关链接具有难度，算法依然具有良好的鲁棒性。

更简单一点说，如果你又一个图或者网络，并想理解其中元素的相对重要性，优先性，排名或者相关性，可以用PageRank试一试。

哪里使用过它呢？Google 拥有PageRank 的商标。但是斯坦福大学取得了PageRank 算法的专利权。如果使用 PageRank，你可能会有疑问： 我不是律师，所以最好和一个真正的律师确认一下。但是只要和 Google 或斯坦福没有涉及到商业竞争，应该都是可以使用这个算法的。

给出PageRank 的三个实现：

1 C++ OpenSource PageRank Implementation

2 Python PageRank Implementation

3 igraph – The network analysis package (R)

**2.AdaBoost 迭代算法**

AdaBoost 算法是做什么的？AdaBoost 是个构建分类器的提升算法。

也许你还记得，分类器拿走大量数据，并试图预测或者分类新数据元素的属于的类别。

但是，提升(boost) 指的什么？提升是个处理多个学习算法（比如决策树）并将他们合并联合起来的综合的学习算法。目的是将弱学习算法综合或形成一个组，把他们联合起来创造一个新的强学习器。

强弱学习器之间有什么区别呢？弱学习分类器的准确性仅仅比猜测高一点。一个比较流行的弱分类器的例子就是只有一层的决策树。

另一个，强学习分类器有更高的准确率，一个通用的强学习器的例子就是 SVM。

举个 AdaBoost 算法的例子：我们开始有3个弱学习器，我们将在一个包含病人数据的数据训练集上对他们做10轮训练。数据集里包含了病人的医疗记录各个细节。

问题来了，那我们怎么预测某个病人是否会得癌症呢？AdaBoost 是这样给出答案的：

第一轮，AdaBoost 拿走一些训练数据，然后测试每个学习器的准确率。最后的结果就是我们找到最好的那个学习器。另外，误分类的样本学习器给予一个比较高的权重，这样他们在下轮就有很高的概率被选中了。

再补充一下，最好的那个学习器也要给根据它的准确率赋予一个权重，并将它加入到联合学习器中（这样现在就只有一个分类器了）

第二轮， AdaBoost 再次试图寻找最好的学习器。

关键部分来了，病人数据样本的训练数据现在被有很高误分配率的权重影响着。换句话说，之前误分类的病人在这个样本里有很高的出现概率。

为什么？

这就像是在电子游戏中已经打到了第二级，但当你的角色死亡后却不必从头开始。而是你从第二级开始然后集中注意，尽力升到第三级。

同样地，第一个学习者有可能对一些病人的分类是正确的，与其再度试图对他们分类，不如集中注意尽力处理被误分类的病人。

最好的学习器也被再次赋予权重并加入到联合分类器中，误分类的病人也被赋予权重，这样他们就有比较大的可能性再次被选中，我们会进行过滤和重复。

在10轮结束的时候，我们剩下了一个带着不同权重的已经训练过的联合学习分类器，之后重复训练之前回合中被误分类的数据。

这是个监督还是非监督算法？因为每一轮训练带有已经标记好数据集的弱训练器，因此这是个监督学习。

为什么使用 AdaBoost？AdaBoost算法简单， 编程相对来说简洁直白。

另外，它速度快！弱学习器 一般都比强学习器简单，简单意味着它们的运行速度可能更快。

还有件事：

因为每轮连续的Adaboost回合都重新定义了每个最好学习器的权重，因此这是个自动调整学习分类器的非常简洁的算法，你所要做的所有事就是指定运行的回合数。

最后，算法灵活通用，AdaBoost 可以加入任何学习算法，并且它能处理多种数据。

AdaBoost 有很多程序实现和变体。给出一些：

▪ scikit-learn

▪ ICSIBoost

▪ gbm: Generalized Boosted Regression Models

如果你喜欢Mr.Rogers,你会喜欢下面的算法的…

**3.kNN：k最近邻算法**

它是做什么的？kNN，或 K 最近邻(k-Nearest Neighbors), 诗歌分类算法。然而，它和我们之前描述的分类器不同，因为它是个懒散学习法。

什么是懒散学习法呢？和存储训练数据的算法不同，懒散学习法在训练过程中不需要做许多处理。只有当新的未被分类的数据输入时，这类算法才会去做分类。

但在另一方面，积极学习法则会在训练中建立一个分类模型，当新的未分类数据输入时，这类学习器会把新数据也提供给这个分类模型。

那么 C4.5，SVM 和 AdaBoost 属于哪类呢？不像 kNN算法，他们都是积极学习算法。

给出原因：

1 C4.5 在训练中建立了一个决策分类树模型。

2 SVM在训练中建立了一个超平面的分类模型。

3 AdaBoost在训练中建立了一个联合的分类模型。

那么 kNN 做了什么？ kNN 没有建立这样的分类模型，相反，它只是储存了一些分类好的训练数据。那么新的训练数据进入时，kNN 执行两个基本步骤：

1 首先，它观察最近的已经分类的训练数据点—也就是，k最临近点（k-nearest neighbors）

2 第二部，kNN使用新数据最近的邻近点的分类， 就对新数据分类得到了更好的结果了。

你可能会怀疑…kNN 是怎么计算出最近的是什么？ 对于连续数据来说，kNN 使用一个像欧氏距离的距离测度，距离测度的选择大多取决于数据类型。有的甚至会根据训练数据学习出一种距离测度。关于 kNN 距离测度有更多的细节讨论和论文描述。

对于离散数据，解决方法是可以把离散数据转化为连续数据。给出两个例子：

1 使用汉明距离（Hamming distance ）作为两个字符串紧密程度的测度。

2 把离散数据转化为二进制表征。

这两个来自Stack Overflow的思路也有一些关于处理离散数据的建议：

▪ KNN classification with categorical data

▪ Using k-NN in R with categorical values

当临近的点是不同的类，kNN 怎么给新数据分类呢？当临近点都是同一类的时候，kNN 也就不费力气了。我们用直觉考虑，如果附近点都一致，那么新数据点就很可能落入这同一个类中了。

我打赌你能猜到事情是从哪里开始变的麻烦的了…

当临近点不是同一类时，kNN 怎么决定分类情况的呢？

处理这种情况通常有两种办法：

1 通过这些临近点做个简单的多数投票法。哪个类有更多的票，新数据就属于那个类。

2 还是做个类似的投票，但是不同的是，要给那些离的更近的临近点更多的投票权重。这样做的一个简单方法是使用反距离(reciprocal distance). 比如，如果某个临近点距离5个单位，那么它的投票权重就是1/5.当临近点越来越远是，倒数距离就越来越小…这正是我们想要的。

这是个监督算法还是非监督的呢？因为 kNN 算法提供了已经被分类好的数据集，所以它是个监督学习算法。

为什么我们会用 kNN？便于理解和实现是我们使用它的两个关键原因。根据距离测度的方法，kNN 可能会非常精确。

但是这还只是故事的一部分，下面是我们需要注意的5点：

1 当试图在一个大数据集上计算最临近点时，kNN 算法可能会耗费高昂的计算成本。

2 噪声数据(Noisy data)可能会影响到 kNN 的分类。

3 选择大范围的属性筛选(feature)会比小范围的筛选占有很多优势，所以属性筛选(feature)的规模非常重要。

4 由于数据处理会出现延迟，kNN 相比积极分类器，一般需要更强大的存储需求。

5 选择一个合适的距离测度对 kNN 的准确性来说至关重要。

哪里用过这个方法？有很多现存的 kNN 实现手段：

▪ MATLAB k-nearest neighbor classification

▪ scikit-learn KNeighborsClassifier

▪ k-Nearest Neighbour Classification in R

是不是垃圾，先别管了。先读读下面的算法吧….

**4. Naive Bayes 朴素贝叶斯算法**

算法是做什么的？朴素贝叶斯（Naive Bayes）并不只是一个算法，而是一系列分类算法，这些算法以一个共同的假设为前提：

被分类的数据的每个属性与在这个类中它其他的属性是独立的。

独立是什么意思呢？当一个属性值对另一个属性值不产生任何影响时，就称这两个属性是独立的。

举个例子：

比如说你有一个病人的数据集，包含了病人的脉搏，胆固醇水平，体重，身高和邮编这样的属性。如果这些属性值互相不产生影响，那么所有属性都是独立的。对于这个数据集来说，假定病人的身高和邮编相互独立，这是合理的。因为病人的身高和他们的邮编没有任何关系。但是我们不能停在这，其他的属性间是独立的么？

很遗憾，答案是否定的。给出三个并不独立的属性关系：

▪ 如果身高增加，体重可能会增加。

▪ 如果胆固醇水平增加，体重可能增加。

▪ 如果胆固醇水平增加，脉搏也可能会增加。

以我的经验来看，数据集的属性一般都不是独立的。

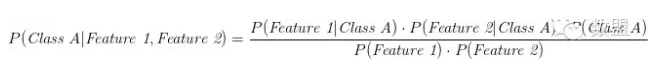
这样就和下面的问题联系起来了…

为什么要把算法称为朴素的(naive)呢？数据集中所有属性都是独立的这个假设正是我们称为朴素（naive）的原因—— 通常下例子中的所有属性并不是独立的。

什么是贝叶斯（Bayes）？Thomas Bayes 是一个英国统计学家，贝叶斯定理就是以他名字命名的。点击这个链接可以知道更多贝叶斯定理的内容（Bayes’ Theorem）

总而言之，根据给定的一系列属性信息，借用概率的知识，我们可以使用这个定理来预测分类情况。

分类的简化等式看起来就像下面的这个式子：



我们在深入研究一下..

这个等式是什么意思？在属性1和属性2的条件下，等式计算出了A 类的概率。换句话说，如果算出属性1 和2，等式算出的数据属于 A 类的概率大小。

等式这样写解释为：在属性1和属性2条件下，分类 A 的概率是一个分数。

▪ 分数的分子是在分类 A条件下属性1的概率，乘以在分类 A 条件下属性2的概率，再乘以分类 A 的概率

▪ 分数的分母是属性1的概率乘以属性2的概率。

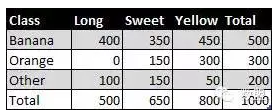
举个 Naive Bayes 的例子，下面是一个从 Stack Overflow thread (Ram’s answer)中找到的一个好例子。

事情是这样的：

▪ 我们有个1000个水果的训练数据集。

▪ 水果可能是香蕉，橘子或者其他（这些水果种类就是类）

▪ 水果可能是长形的、甜的、或者黄颜色的(这些是属性).



在这个训练集中你发现了什么？

▪ 500个香蕉中，长的有400个、甜的有350个、黄色的450个

▪ 300个橘子中、没有长的、甜的150个、黄色的300个

▪ 还剩下的200个水果中、长的100个、甜的150个、黄色的50个

如果我们根据长度、甜度和水果颜色，在不知道它们类别的情况下，我们现在可以计算水果是香蕉、橘子或者其他水果的概率了。

假设我们被告知这个未分类的水果是长的、甜的、黄色的。

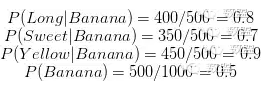
下面我们以4个步骤来计算所有的概率：

第一步：想要计算水果是香蕉的概率，我们首先发现这个式子看起来很熟悉。这就是在属性为长形、甜和黄色的条件下，水果是香蕉类的概率，这个表达更简洁一些：



这确实就像我们之前讨论的那个等式。

第二步：以分子开始，让我们把公式的所有东西都加进去。



像公式一样，把所有的都乘起来，我们就得到了：



第三步：不用管分母了，因为计算别的分类时分子是一样的。

第四步：计算其他类时也做类似的计算：



因为0.252大于0.01875，Naive Bayes 会把长形，甜的还是黄色水果分到香蕉的一类中。

这是个监督算法还是非监督算法呢？ 为了得到频数表，Naive Bayes 提供了已经分好类的训练数据集，所以这是个监督学习算法。

为什么使用 Naive Bayes？就像你在上面看到的例子一样，Naive Bayes 只涉及到了简单的数学知识。加起来只有计数、乘法和除法而已。

一旦计算好了频数表(frequency tables),要分类一个未知的水果只涉及到计算下针对所有类的概率，然后选择概率最大的即可。

尽管算法很简单，但是 Naive Bayes 却出人意料的十分精确。比如，人们发现它是垃圾邮件过滤的高效算法。

Naive Bayes 的实现可以从Orange, scikit-learn, Weka 和 R 里面找到。

**5.C4.5算法**

C4.5是做什么的？C4.5 以决策树的形式构建了一个分类器。为了做到这一点，需要给定 C4.5 表达内容已分类的数据集合。

等下，什么是分类器呢？ 分类器是进行数据挖掘的一个工具，它处理大量需要进行分类的数据，并尝试预测新数据所属的类别。

举个例子吧，假定一个包含很多病人信息的数据集。我们知道每个病人的各种信息，比如年龄、脉搏、血压、最大摄氧量、家族病史等。这些叫做数据属性。

现在：

给定这些属性，我们想预测下病人是否会患癌症。病人可能会进入下面两个分类：会患癌症或者不会患癌症。 C4.5 算法会告诉我们每个病人的分类。

做法是这样的：

用一个病人的数据属性集和对应病人的反馈类型，C4.5 构建了一个基于新病人属性预测他们类型的决策树。

这点很棒，那么什么是决策树呢？决策树学习是创建一种类似与流程图的东西对新数据进行分类。使用同样的病人例子，一个特定的流程图路径可以是这样的：

* 病人有癌症的病史
* 病人有和癌症病人高度相似的基因表达
* 病人有肿瘤
* 病人的肿瘤大小超过了5cm

基本原则是：

流程图的每个环节都是一个关于属性值的问题，并根据这些数值，病人就被分类了。你可以找到很多决策树的例子。

算法是监督学习还是无监督学习呢？这是一个监督学习算法，因为训练数据是已经分好类的。使用分好类的病人数据，C4.5算法不需要自己学习病人是否会患癌症。

那 C4.5 算法和决策树系统有什么区别呢？

首先，C4.5 算法在生成信息树的时候使用了信息增益。

其次，尽管其他系统也包含剪枝，C4.5使用了一个单向的剪枝过程来缓解过渡拟合。剪枝给结果带来了很多改进。

再次，C4.5算法既可以处理连续数据也可以处理离散数据。我的理解是，算法通过对连续的数据指定范围或者阈值，从而把连续数据转化为离散的数据。

最后，不完全的数据用算法自有的方式进行了处理。

为什么使用 C4.5算法呢？可以这么说，决策树最好的卖点是他们方便于翻译和解释。他们速度也很快，是种比较流行的算法。输出的结果简单易懂。

哪里可以使用它呢？ 在 OpenTox 上可以找到一个很流行的开源 Java实现方法。Orange 是一个用于数据挖掘的开源数据可视化和分析工具，它的决策树分类器是用 C4.5实现的。

分类器是很棒的东西，但也请看看下一个聚类算法….

英文出处：rayli

除了上述的10种算法，这里还总结了其他常用的挖掘算法，也许会有你想要的：

**1.FP-Tree(频繁模式树)算法。**这个算法也有被称为FP-growth算法，这个算法克服了Apriori算法的产生过多侯选集的缺点，通过递归的产生频度模式树，然后对树进行挖掘，后面的过程与Apriori算法一致。

**2.HITS算法**。HITS算法是另外一个链接算法，部分原理与PageRank算法是比较相似的，HITS算法引入了权威值和中心值的概念，HITS算法是受用户查询条件影响的，他一般用于小规模的数据链接分析，也更容易遭受到攻击。

**3.BIRCH算法。**BIRCH算法利用构建CF聚类特征树作为算法的核心，通过树的形式，BIRCH算法扫描数据库，在内存中建立一棵初始的CF-树，可以看做数据的多层压缩。

**4.GSP算法。**GSP算法是序列模式挖掘算法。GSP算法也是Apriori类算法，在算法的过程中也会进行连接和剪枝操作，不过在剪枝判断的时候还加上了一些时间上的约束等条件。

**5.PreFixSpan算法。**PreFixSpan算法是另一个序列模式挖掘算法，在算法的过程中不会产生候选集，给定初始前缀模式，不断的通过后缀模式中的元素转到前缀模式中，而不断的递归挖掘下去。

**6.CBA(基于关联规则分类)算法。**CBA算法是一种集成挖掘算法，因为他是建立在关联规则挖掘算法之上的，在已有的关联规则理论前提下，做分类判断，只是在算法的开始时对数据做处理，变成类似于事务的形式。

**7.RoughSets(粗糙集)算法。**粗糙集理论是一个比较新颖的数据挖掘思想。这里使用的是用粗糙集进行属性约简的算法，通过上下近似集的判断删除无效的属性，进行规制的输出。

**8.gSpan算法。**gSpan算法属于图挖掘算法领域。主要用于频繁子图的挖掘，相较于其他的图算法，子图挖掘算法是他们的一个前提或基础算法。gSpan算法用到了DFS编码，和Edge五元组，最右路径子图扩展等概念，算法比较的抽象和复杂。

## ****6. k 均值聚类算法****

它是做什么的呢？K-聚类算法从一个目标集中创建多个组，每个组的成员都是比较相似的。这是个想要探索一个数据集时比较流行的聚类分析技术。

等下，什么是聚类分析呢？聚类分析属于设计构建组群的算法，这里的组成员相对于非组成员有更多的相似性。在聚类分析的世界里，类和组是相同的意思。

举个例子，假设我们定义一个病人的数据集。在聚类分析里，这些病人可以叫做观察对象。我们知道每个病人的各类信息，比如年龄、血压、血型、最大含氧量和胆固醇含量等。这是一个表达病人特性的向量。

请看：

你可以基本认为一个向量代表了我们所知道的病人情况的一列数据。这列数据也可以理解为多维空间的坐标。脉搏是一维坐标，血型是其他维度的坐标等等。

你可能会有疑问：

给定这个向量集合，我们怎么把具有相似年龄、脉搏和血压等数据的病人聚类呢？

想知道最棒的部分是什么吗？

你告诉 k-means 算法你想要多少种类。K-means 算法会处理后面的部分。

那它是怎么处理的呢？k-means 算法有很多优化特定数据类型的变量。

Kmeans算法更深层次的这样处理问题：

1. k-means 算法在多维空间中挑选一些点代表每一个 k 类。他们叫做中心点。
2. 每个病人会在这 k 个中心点中找到离自己最近的一个。我们希望病人最靠近的点不要是同一个中心点，所以他们在靠近他们最近的中心点周围形成一个类。
3. 我们现在有 k 个类，并且现在每个病人都是一个类中的一员。
4. 之后k-means 算法根据它的类成员找到每个 k 聚类的中心（没错，用的就是病人信息向量）
5. 这个中心成为类新的中心点。
6. 因为现在中心点在不同的位置上了，病人可能现在靠近了其他的中心点。换句话说，他们可能会修改自己的类成员身份。
7. 重复2-6步直到中心点不再改变，这样类成员也就稳定了。这也叫做收敛性。

这算法是监督的还是非监督的呢？这要看情况了，但是大多数情况下 k-means 会被划分为非监督学习的类型。并不是指定分类的个数，也没有观察对象该属于那个类的任何信息，k-means算法自己“学习”如何聚类。k-means 可以是半监督的。

为什么要使用 k-means 算法呢？我认为大多数人都同意这一点：

k-means 关键卖点是它的简单。它的简易型意味着它通常要比其他的算法更快更有效，尤其是要大量数据集的情况下更是如此。

他可以这样改进：

k-means 可以对已经大量数据集进行预先聚类处理，然后在针对每个子类做成本更高点的聚类分析。k-means 也能用来快速的处理“K”和探索数据集中是否有被忽视的模式或关系。

但用k-means 算法也不是一帆风顺的：

k means算法的两个关键弱点分别是它对异常值的敏感性和它对初始中心点选择的敏感性。最后一个需要记住的是， K-means 算法是设计来处理连续数据的。对于离散数据你需要使用一些小技巧后才能让 K-means 算法奏效。

Kmeans 在哪里使用过呢？ 网上有很多可获得的 kmeans 聚类算法的语言实现：

▪ Apache Mahout

▪ Julia

▪ R

▪ SciPy

▪ Weka

▪ MATLAB

▪ SAS

如果决策树和聚类算法还没有打动你，那么你会喜欢下一个算法的。

## ****7.支持向量机****

它是做什么的呢？支持向量机（SVM）获取一个超平面将数据分成两类。以高水准要求来看，除了不会使用决策树以外，SVM与 C4.5算法是执行相似的任务的。

咦？一个超..什么？ 超平面（hyperplane）是个函数，类似于解析一条线的方程。实际上，对于只有两个属性的简单分类任务来说，超平面可以是一条线的。

其实事实证明：

SVM 可以使用一个小技巧，把你的数据提升到更高的维度去处理。一旦提升到更高的维度中，SVM算法会计算出把你的数据分离成两类的最好的超平面。

有例子么？当然，举个最简单的例子。我发现桌子上开始就有一堆红球和蓝球，如果这这些球没有过分的混合在一起，不用移动这些球，你可以拿一根棍子把它们分离开。

你看，当在桌上加一个新球时，通过已经知道的棍字的哪一边是哪个颜色的球，你就可以预测这个新球的颜色了。

最酷的部分是什么呢？SVM 算法可以算出这个超平面的方程。

如果事情变得更复杂该怎么办？当然了，事情通常都很复杂。如果球是混合在一起的，一根直棍就不能解决问题了。

下面是解决方案：

快速提起桌子，把所有的球抛向空中，当所有的球以正确的方式抛在空中是，你使用一张很大的纸在空中分开这些球。

你可能会想这是不是犯规了。不，提起桌子就等同于把你的数据映射到了高维空间中。这个例子中，我们从桌子表面的二维空间过度到了球在空中的三维空间。

那么 SVM该怎么做呢？通过使用核函数（kernel），我们在高维空间也有很棒的操作方法。这张大纸依然叫做超平面，但是现在它对应的方程是描述一个平面而不是一条线了。根据 Yuval 的说法，一旦我们在三维空间处理问题，超平面肯定是一个面而不是线了。

关于 SVM的解释思路，Reddit 的 ELI5 和 ML 两个子版块上也有两个很棒的讨论帖。

那么在桌上或者空中的球怎么用现实的数据解释呢？桌上的每个球都有自己的位置，我们可以用坐标来表示。打个比方，一个球可能是距离桌子左边缘20cm 距离底部边缘 50 cm，另一种描述这个球的方式是使用坐标(x,y)或者(20,50)表达。x和 y 是代表球的两个维度。

可以这样理解：如果我们有个病人的数据集，每个病人可以用很多指标来描述，比如脉搏，胆固醇水平，血压等。每个指标都代表一个维度。

基本上，SVM 把数据映射到一个更高维的空间然后找到一个能分类的超平面。

类间间隔(margin)经常会和 SVM 联系起来，类间间隔是什么呢？它是超平面和各自类中离超平面最近的数据点间的距离。在球和桌面的例子中，棍子和最近的红球和蓝球间的距离就是类间间隔(margin)。

SVM 的关键在于，它试图最大化这个类间间隔，使分类的超平面远离红球和蓝球。这样就能降低误分类的可能性。

那么支持向量机的名字是哪里来的？还是球和桌子的例子中，超平面到红球和蓝球的距离是相等的。这些球或者说数据点叫做支持向量，因为它们都是支持这个超平面的。

那这是监督算法还是非监督的呢？SVM 属于监督学习。因为开始需要使用一个数据集让 SVM学习这些数据中的类型。只有这样之后 SVM 才有能力对新数据进行分类。

为什么我们要用 SVM 呢？ SVM 和 C4.5大体上都是优先尝试的二类分类器。根据“没有免费午餐原理”，没有哪一种分类器在所有情况下都是最好的。此外，核函数的选择和可解释性是算法的弱点所在。

在哪里使用 SVM？有什么 SVM 的实现方法，比较流行的是用scikit-learn, MATLAB 和 libsvm实现的这几种。

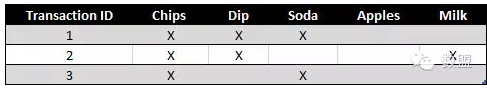
下面要介绍的算法是我最喜欢的算法之一：

## ****8. Apriori 关联算法****

它是做什么的？Apriori算法学习数据的关联规则(association rules)，适用于包含大量事务（transcation）的数据库。

什么是关联规则？关联规则学习是学习数据库中不同变量中的相互关系的一种数据挖掘技术。

举个 Apriori 算法的例子：我们假设有一个充满超市交易数据的数据库，你可以把数据库想象成一个巨大的电子数据表，表里每一行是一个顾客的交易情况，每一列代表不用的货物项。



精彩的部分来了：通过使用 Apriori 算法，我们就知道了同时被购买的货物项，这也叫做关联规则。它的强大之处在于，你能发现相比较其他货物来说，有一些货物更频繁的被同时购买—终极目的是让购物者买更多的东西。这些常被一起购买的货物项被称为项集（itemset）。

举个例子，你大概能很快看到“薯条+蘸酱”和“薯条+苏打水”的组合频繁的一起出现。这些组合被称为2-itemsets。在一个足够大的数据集中，就会很难“看到”这些关系了，尤其当还要处理3-itemset 或者更多项集的时候。这正是 Apriori 可以帮忙的地方！

你可能会对 Apriori 算法如何工作有疑问，在进入算法本质和细节之前，得先明确3件事情：

1. 第一是你的项集的大小，你想看到的模式是2-itemset或3-itemset 还是其他的？
2. 第二是你支持的项集，或者是从事务的总数划分出的事务包含的项集。一个满足支持度的项集叫做频繁项集。
3. 第三是根据你已经统计的项集中某些数据项，计算其他某个数据项出现的信心水准或是条件概率。例如项集中出现的薯片的话，有67%的信心水准这个项集中也会出现苏打水。

基本的 Apriori 算法有三步：

1. 参与，扫描一遍整个数据库，计算1-itemsets 出现的频率。
2. 剪枝，满足支持度和可信度的这些1-itemsets移动到下一轮流程，再寻找出现的2-itemsets。
3. 重复，对于每种水平的项集 一直重复计算，知道我们之前定义的项集大小为止。

这个算法是监督的还是非监督的？Apriori 一般被认为是一种非监督的学习方法，因为它经常用来挖掘和发现有趣的模式和关系。

但是，等下，还有呢…对Apriori 算法改造一下也能对已经标记好的数据进行分类。

为什么使用Apriori 算法？它易于理解，应用简单，还有很多的派生算法。

但另一方面…

当生成项集的时候，算法是很耗费内存、空间和时间。

大量的 Apriori 算法的语言实现可供使用。比较流行的是 ARtool, Weka, and Orange。

下一个算法对我来说是最难的，一起来看下吧。

## ****9.EM 最大期望算法 Expectation Maximization****

EM 算法是做什么的？在数据挖掘领域，最大期望算法（Expectation-Maximization,EM） 一般作为聚类算法（类似 kmeans 算法）用来知识挖掘。

在统计学上，当估算带有无法观测隐藏变量的统计模型参数时,EM 算法不断迭代和优化可以观测数据的似然估计值。

好，稍等让我解释一下…

我不是一个统计学家，所以希望我的简洁表达能正确并能帮助理解。

下面是一些概念，能帮我们更好的理解问题。

什么事统计模型？我把模型看做是描述观测数据是如何生成的。例如，一场考试的分数可能符合一种钟形曲线，因此这种分数分布符合钟形曲线（也称正态分布）的假设就是模型。

等下，那什么是分布？分布代表了对所有可测量结果的可能性。例如，一场考试的分数可能符合一个正态分布。这个正态分布代表了分数的所有可能性。换句话说，给定一个分数，你可以用这个分布来预计多少考试参与者可能会得到这个分数。

这很不错，那模型的参数又是什么呢？作为模型的一部分，分布属性正是由参数来描述的。例如，一个钟形曲线可以用它的均值和方差来描述。

还是使用考试的例子，一场考试的分数分布（可测量的结果）符合一个钟形曲线（就是分布）。均值是85，方差是100.

那么，你描述正态分布需要的所有东西就是这两个参数：

1. 平均值
2. 方差

那么，似然性呢？回到我们之前的钟形曲线例子，假设我们已经拿到很多的分数数据，并被告知分数符合一个钟形曲线。然而，我们并没有给到所有的分数，只是拿到了一个样本。

可以这样做：

我们不知道所有分数的平均值或者方差，但是我们可以使用样本计算它们。似然性就是用估计的方差和平均值得到的钟形曲线在算出很多分数的概率。

换句话说，给定一系列可测定的结果，让我们来估算参数。再使用这些估算出的参数，得到结果的这个假设概率就被称为似然性。

记住，这是已存在分数的假设概率，并不是未来分数的概率。

你可能会疑问，那概率又是什么？

还用钟形曲线的例子解释，假设我们知道均值和方差。然我们被告知分数符合钟形曲线。我们观察到的某些分数的可能性和他们多久一次的被观测到就是概率。

更通俗的讲，给定参数，让我们来计算可以观察到什么结果。这就是概率为我们做的事情。

很好，现在，观测到的数据和未观测到的隐藏数据区别在哪里？观测到的数据就是你看到或者记录的数据。未观测的数据就是遗失的数据。数据丢失的原因有很多（没有记录，被忽视了，等等原因）。

算法的优势是：对于数据挖掘和聚类，观察到遗失的数据的这类数据点对我们来说很重要。我们不知道具体的类，因此这样处理丢失数据对使用 EM 算法做聚类的任务来说是很关键的。

再说一次，当估算带有无法观测隐藏变量的统计模型参数时,EM 算法不断迭代和优化可以观测数据的似然估计值。 希望现在再说更容易理解了。

算法的精髓在于：

通过优化似然性，EM 生成了一个很棒的模型，这个模型可以对数据点指定类型标签—听起来像是聚类算法！

EM 算法是怎么帮助实现聚类的呢？EM 算法以对模型参数的猜测开始。然后接下来它会进行一个循环的3步：

1. E 过程：基于模型参数，它会针对每个数据点计算对聚类的分配概率。
2. M 过程：基于 E 过程的聚类分配，更新模型参数。
3. 重复知道模型参数和聚类分配工作稳定（也可以称为收敛）。

EM 是监督算法还是非监督算法呢？因为我们不提供已经标好的分类信息，这是个非监督学习算法。

为什么使用它？EM 算法的一个关键卖点就是它的实现简单直接。另外，它不但可以优化模型参数，还可以反复的对丢失数据进行猜测。

这使算法在聚类和产生带参数的模型上都表现出色。在得知聚类情况和模型参数的情况下，我们有可能解释清楚有相同属性的分类情况和新数据属于哪个类之中。

不过EM 算法也不是没有弱点…

第一，EM 算法在早期迭代中都运行速度很快，但是越后面的迭代速度越慢。

第二，EM 算法并不能总是寻到最优参数，很容易陷入局部最优而不是找到全局最优解。

EM 算法实现可以在 Weka中找到，mclust package里面有 R 语言对算法的实现，scikit-learn的gmm module里也有对它的实现。

最后，看一下第十种算法吧。

## ****10.CART 分类算法****

算法是做什么的？ CART 代表分类和回归树(classification and regression trees)。它是个决策树学习方法，同时输出分类和回归树。 像 C4.5一样，CART 是个分类器。

分类树像决策树一样么？分类树是决策树的一种。分类树的输出是一个类。

举个例子，根据一个病人的数据集、你可能会试图预测下病人是否会得癌症。这个分类或者是“会的癌症”或者是“不会得癌症”。

那回归树是什么呢？和分类树预测分类不同，回归树预测一个数字或者连续数值，比如一个病人的住院时间或者一部智能手机的价格。

这么记比较简单：

分类树输出类、回归树输出数字。

由于我们已经讲过决策树是如何分类数据的了，我们就直接跳过进入正题了…

CART和 C4.5对比如下：



这是个监督算法还是非监督的呢？为了构造分类和回归树模型，需要给它提供被分类好的训练数据集，因此 CART 是个监督学习算法。

为什么要使用 CART 呢？使用 C4.5的原因大部分也适用于 CART，因为它们都是决策树学习的方法。便于说明和解释这类的原因也适用于 CART。

和 C4.5一样，它们的计算速度都很快，算法也都比较通用流行，并且输出结果也具有可读性。

scikit-learn 在他们的决策树分类器部分实现了 CART 算法；R 语言的 tree package 也有 CART 的实现；Weka 和 MATLAB 也有CART的实现过程。

最后，基于斯坦福和加州大学伯克利分校的世界闻名的统计学家们的理论，只有 Salford系统有最原始的 CART 专利源码的实现部分。