**1 回归问题**

回归在数学上来说是给定一个点集，能够找一条曲线去拟合。  
这里面的“找一条”曲线不是漫无边际的找，而是先假定曲线的形式，如：直线、二次曲线等等，然后来学习确定曲线的各项参数。

一方面，[**算法**](http://lib.csdn.net/base/datastructure)没有那么神奇，不能够告诉我们用什么类型的曲线拟合最好；另一方面，如果最初的假定就错了（用直线拟合最好，但是认为选择了二次曲线），  
最终的效果也是不好的。

**所以回归问题，人的经验是是很重要的。要通过人分析数据，总结曲线的形式。**

**2 线性回归**

就是通过学习一条直线来拟合样本点。在二维平面上，直线的形式为 y = ax + b，就是学习参数a和b。在N维平面上，形式为 y = a1x1 + a2x2 + ...... + b，学习a1...an-1和b  
  
（当然，y也可以是x的非线性函数，如：y = a \* x^2，非线性回归的处理方式和线性回归类似）  
  
在现实中，这种模型有缺点：  
xi是模型输入，在现实中，模型的输入多种多样，连续值、离散值、枚举值；而且输入的范围差别很大。如：室外温度的范围[-50, 50]，某概率的范围[0, 1]，等等。

数值大的输入在实际计算中往往就使得输入范围小的输入的作用可以忽略不计了。而且，这还是拟合问题，在实际中分类问题更有价值。这就产生了3，逻辑回归模型。

# Logistic Regression 模型简介

FIN ·2015-05-08 10:00

逻辑回归（Logistic Regression）是机器学习中的一种分类模型，由于算法的简单和高效，在实际中应用非常广泛。本文作为美团机器学习InAction系列中的一篇，主要关注逻辑回归算法的数学模型和参数求解方法，最后也会简单讨论下逻辑回归和贝叶斯分类的关系，以及在多分类问题上的推广。

## 逻辑回归

### 问题

实际工作中，我们可能会遇到如下问题：

1. 预测一个用户是否点击特定的商品
2. 判断用户的性别
3. 预测用户是否会购买给定的品类
4. 判断一条评论是正面的还是负面的

这些都可以看做是分类问题，更准确地，都可以看做是二分类问题。同时，这些问题本身对美团也有很重要的价值，能够帮助我们更好的了解我们的用户，服务我们的用户。要解决这些问题，通常会用到一些已有的分类算法，比如逻辑回归，或者支持向量机。它们都属于有监督的学习，因此在使用这些算法之前，必须要先收集一批标注好的数据作为训练集。有些标注可以从log中拿到（用户的点击，购买），有些可以从用户填写的信息中获得（性别），也有一些可能需要人工标注（评论情感极性）。另一方面，知道了一个用户或者一条评论的标签后，我们还需要知道用什么样的特征去描述我们的数据，对用户来说，可以从用户的浏览记录和购买记录中获取相应的统计特征，而对于评论来说，最直接的则是文本特征。这样拿到数据的特征和标签后，就得到一组训练数据：

D=(x1,y1),(x2,y2)...(xN,yN)D=(x1,y1),(x2,y2)...(xN,yN)

其中 xixi 是一个 mm 维的向量，xi=[xi1,xi2,...,xim]xi=[x1i,x2i,...,xmi] ，yy 在 {0, 1} 中取值。（本文用{1，0}表示正例和负例，后文沿用此定义。）

我们的问题可以简化为，如何找到这样一个决策函数y∗=f(x)y∗=f(x)，它在未知数据集上能有足够好的表现。至于如何衡量一个二分类模型的好坏，我们可以用分类错误率这样的指标：Err=1N∑1[y∗=y]Err=1N∑1[y∗=y] 。也可以用准确率，召回率，AUC等指标来衡量。

值得一提的是，模型效果往往和所用特征密切相关。特征工程在任何一个实用的机器学习系统中都是必不可少的，机器学习InAction系列已有一篇文章中对此做了详细的介绍，本文不再详细展开。

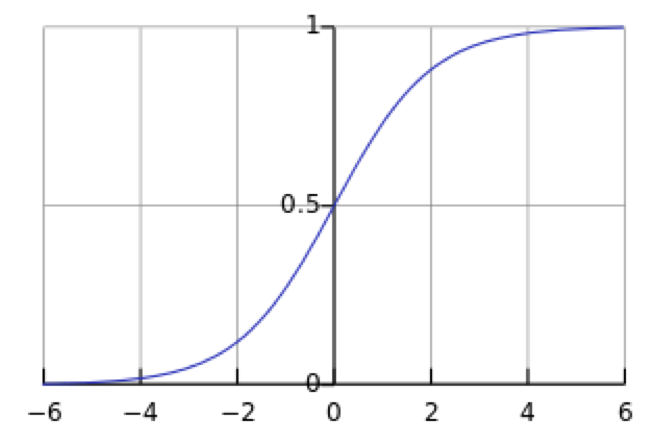
### 模型

#### sigmoid 函数

在介绍逻辑回归模型之前，我们先引入sigmoid函数，其数学形式是：

g(x)=11+e−xg(x)=11+e−x

对应的函数曲线如下图所示：



从上图可以看到sigmoid函数是一个s形的曲线，它的取值在[0, 1]之间，在远离0的地方函数的值会很快接近0/1。这个性质使我们能够以概率的方式来解释（后边延伸部分会简单讨论为什么用该函数做概率建模是合理的)。

#### 决策函数

一个机器学习的模型，实际上是把决策函数限定在某一组条件下，这组限定条件就决定了模型的假设空间。当然，我们还希望这组限定条件简单而合理。而逻辑回归模型所做的假设是：

P(y=1|x;θ)=g(θTx)=11+e−θT∗xP(y=1|x;θ)=g(θTx)=11+e−θT∗x

这里的 g(h)g(h) 是上边提到的 sigmoid 函数，相应的决策函数为：

y∗=1,ifP(y=1|x)>0.5y∗=1,ifP(y=1|x)>0.5

选择0.5作为阈值是一个一般的做法，实际应用时特定的情况可以选择不同阈值，如果对正例的判别准确性要求高，可以选择阈值大一些，对正例的召回要求高，则可以选择阈值小一些。

#### 参数求解

模型的数学形式确定后，剩下就是如何去求解模型中的参数。统计学中常用的一种方法是最大似然估计，即找到一组参数，使得在这组参数下，我们的数据的似然度（概率）越大。在逻辑回归模型中，似然度可表示为：

L(θ)=P(D|θ)=∏P(y|x;θ)=∏g(θTx)y(1−g(θTx))1−yL(θ)=P(D|θ)=∏P(y|x;θ)=∏g(θTx)y(1−g(θTx))1−y

取对数可以得到对数似然度：

l(θ)=∑ylogg(θTx)+(1−y)log(1−g(θTx))l(θ)=∑ylog⁡g(θTx)+(1−y)log⁡(1−g(θTx))

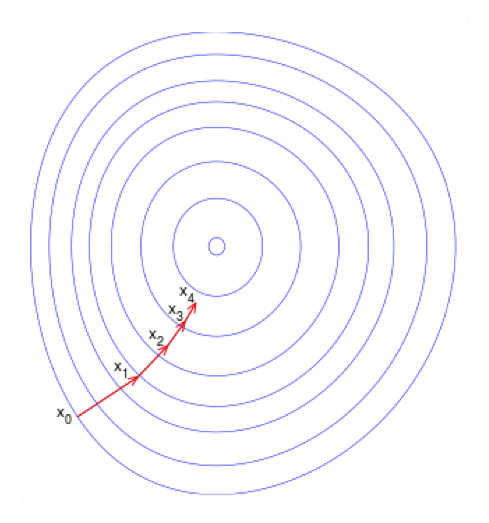
另一方面，在机器学习领域，我们更经常遇到的是损失函数的概念，其衡量的是模型预测错误的程度。常用的损失函数有0-1损失，log损失，hinge损失等。其中log损失在单个数据点上的定义为−ylogp(y|x)−(1−y)log1−p(y|x)−ylog⁡p(y|x)−(1−y)log⁡1−p(y|x)

如果取整个数据集上的平均log损失，我们可以得到

J(θ)=−1Nl(θ)J(θ)=−1Nl(θ)

即在逻辑回归模型中，我们最大化似然函数和最小化log损失函数实际上是等价的。对于该优化问题，存在多种求解方法，这里以梯度下降的为例说明。梯度下降(Gradient Descent)又叫作最速梯度下降，是一种迭代求解的方法，通过在每一步选取使目标函数变化最快的一个方向调整参数的值来逼近最优值。基本步骤如下：

* 选择下降方向（梯度方向，∇J(θ)∇J(θ)）
* 选择步长，更新参数 θi=θi−1−αi∇J(θi−1)θi=θi−1−αi∇J(θi−1)
* 重复以上两步直到满足终止条件



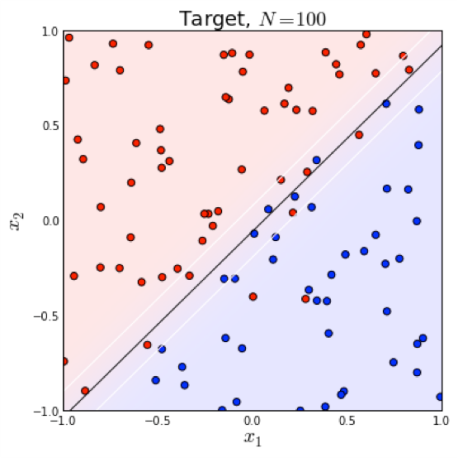
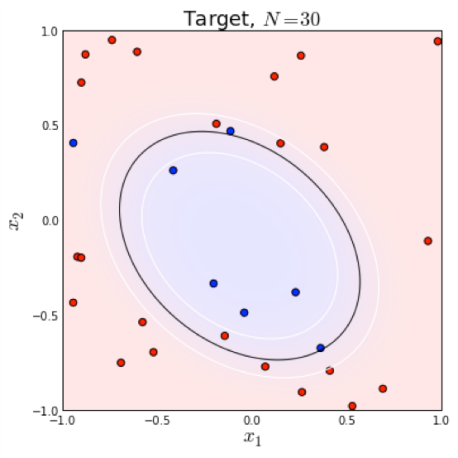
其中损失函数的梯度计算方法为：

∂J∂θ=−1n∑i(yi−y∗i)xi+λθ∂J∂θ=−1n∑i(yi−yi∗)xi+λθ

沿梯度负方向选择一个较小的步长可以保证损失函数是减小的，另一方面，逻辑回归的损失函数是凸函数（加入正则项后是严格凸函数），可以保证我们找到的局部最优值同时是全局最优。此外，常用的凸优化的方法都可以用于求解该问题。例如共轭梯度下降，牛顿法，LBFGS等。

#### 分类边界

知道如何求解参数后，我们来看一下模型得到的最后结果是什么样的。很容易可以从sigmoid函数看出，当θTx>0θTx>0 时，y=1y=1，否则 y=0y=0。θTx=0θTx=0 是模型隐含的分类平面（在高维空间中，我们说是超平面）。所以说逻辑回归本质上是一个线性模型，但是，这不意味着只有线性可分的数据能通过LR求解，实际上，我们可以通过特征变换的方式把低维空间转换到高维空间，而在低维空间不可分的数据，到高维空间中线性可分的几率会高一些。下面两个图的对比说明了线性分类曲线和非线性分类曲线（通过特征映射）。

左图是一个线性可分的数据集，右图在原始空间中线性不可分，但是在特征转换 [x1,x2]=>[x1,x2,x21,x22,x1x2][x1,x2]=>[x1,x2,x12,x22,x1x2] 后的空间是线性可分的，对应的原始空间中分类边界为一条类椭圆曲线。

#### 正则化

当模型的参数过多时，很容易遇到过拟合的问题。这时就需要有一种方法来控制模型的复杂度，典型的做法在优化目标中加入正则项，通过惩罚过大的参数来防止过拟合：

J(θ)=−1N∑ylogg(θTx)+(1−y)log(1−g(θTx))+λ∥w∥pJ(θ)=−1N∑ylog⁡g(θTx)+(1−y)log⁡(1−g(θTx))+λ‖w‖p

一般情况下，取p=1p=1或p=2p=2，分别对应L1，L2正则化，两者的区别可以从下图中看出来，L1正则化（左图）倾向于使参数变为0，因此能产生稀疏解。

实际应用时，由于我们数据的维度可能非常高，L1正则化因为能产生稀疏解，使用的更为广泛一些。

## 延伸

### 生成模型和判别模型

逻辑回归是一种判别模型，表现为直接对条件概率P(y|x)建模，而不关心背后的数据分布P(x,y)。而高斯贝叶斯模型（Gaussian Naive Bayes）是一种生成模型，先对数据的联合分布建模，再通过贝叶斯公式来计算样本属于各个类别的后验概率，即：

p(y|x)=P(x|y)P(y)∑P(x|y)P(y)p(y|x)=P(x|y)P(y)∑P(x|y)P(y)

通常假设P(x|y)是高斯分布，P(y)是多项式分布，相应的参数都可以通过最大似然估计得到。如果我们考虑二分类问题，通过简单的变化可以得到：

logP(y=1|x)P(y=0|x)=logP(x|y=1)P(x|y=0)+logP(y=1)P(y=0) =−(x−μ1)22σ21+(x−μ0)22σ20 +θ0log⁡P(y=1|x)P(y=0|x)=log⁡P(x|y=1)P(x|y=0)+log⁡P(y=1)P(y=0) =−(x−μ1)22σ12+(x−μ0)22σ02 +θ0

如果 σ1=σ0σ1=σ0，二次项会抵消，我们得到一个简单的线性关系：

logP(y=1|x)P(y=0|x)=θTxlog⁡P(y=1|x)P(y=0|x)=θTx

由上式进一步可以得到：

P(y=1|x)=eθTx1+eθTx=11+e−θTxP(y=1|x)=eθTx1+eθTx=11+e−θTx

可以看到，这个概率和逻辑回归中的形式是一样的。这种情况下GNB 和 LR 会学习到同一个模型。实际上，在更一般的假设（P(x|y)的分布属于指数分布族）下，我们都可以得到类似的结论。

### 多分类（softmax)

如果yy不是在[0,1]中取值，而是在KK个类别中取值，这时问题就变为一个多分类问题。有两种方式可以出处理该类问题：一种是我们对每个类别训练一个二元分类器（One-vs-all），当KK个类别不是互斥的时候，比如用户会购买哪种品类，这种方法是合适的。如果KK个类别是互斥的，即 y=iy=i 的时候意味着 yy 不能取其他的值，比如用户的年龄段，这种情况下 Softmax 回归更合适一些。Softmax 回归是直接对逻辑回归在多分类的推广，相应的模型也可以叫做多元逻辑回归（Multinomial Logistic Regression）。模型通过 softmax 函数来对概率建模，具体形式如下：

P(y=i|x,θ)=eθTix∑KjeθTjxP(y=i|x,θ)=eθiTx∑jKeθjTx

而决策函数为：y∗=argmaxiP(y=i|x,θ)y∗=argmaxiP(y=i|x,θ)

对应的损失函数为：

J(θ)=−1N∑iN∑jK1[yi=j]logeθTix∑eθTkxJ(θ)=−1N∑iN∑jK1[yi=j]log⁡eθiTx∑eθkTx

类似的，我们也可以通过梯度下降或其他高阶方法来求解该问题，这里不再赘述。

## 应用

本文开始部分提到了几个在实际中遇到的问题，这里以预测用户对品类的购买偏好为例，介绍一下美团是如何用逻辑回归解决工作中问题的。该问题可以转换为预测用户在未来某个时间段是否会购买某个品类，如果把会购买标记为1，不会购买标记为0，就转换为一个二分类问题。我们用到的特征包括用户在美团的浏览，购买等历史信息，见下表

| **类别** | **特征** |
| --- | --- |
| 用户 | 购买频次，浏览频次，时间，地理位置 ... |
| 品类 | 销量，购买用户，浏览用户 ... |
| 交叉 | 购买频次，浏览频次，购买间隔 ... |

其中提取的特征的时间跨度为30天，标签为2天。生成的训练数据大约在7000万量级（美团一个月有过行为的用户），我们人工把相似的小品类聚合起来，最后有18个较为典型的品类集合。如果用户在给定的时间内购买某一品类集合，就作为正例。哟了训练数据后，使用Spark版的LR算法对每个品类训练一个二分类模型，迭代次数设为100次的话模型训练需要40分钟左右，平均每个模型2分钟，测试集上的AUC也大多在0.8以上。训练好的模型会保存下来，用于预测在各个品类上的购买概率。预测的结果则会用于推荐等场景。

由于不同品类之间正负例分布不同，有些品类正负例分布很不均衡，我们还尝试了不同的采样方法，最终目标是提高下单率等线上指标。经过一些参数调优，品类偏好特征为推荐和排序带来了超过1%的下单率提升。

此外，由于LR模型的简单高效，易于实现，可以为后续模型优化提供一个不错的baseline，我们在排序等服务中也使用了LR模型。

## 总结

逻辑回归的数学模型和求解都相对比较简洁，实现相对简单。通过对特征做离散化和其他映射，逻辑回归也可以处理非线性问题，是一个非常强大的分类器。因此在实际应用中，当我们能够拿到许多低层次的特征时，可以考虑使用逻辑回归来解决我们的问题。

# 

# 3.2、决策树引导

      通俗来说，决策树分类的思想类似于找对象。现想象一个女孩的母亲要给这个女孩介绍男朋友，于是有了下面的对话：

      女儿：多大年纪了？

      母亲：26。

      女儿：长的帅不帅？

      母亲：挺帅的。

      女儿：收入高不？

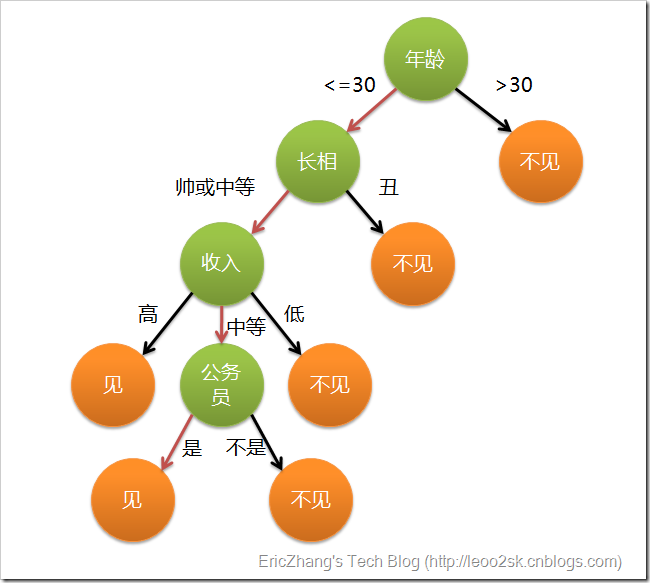
      母亲：不算很高，中等情况。

      女儿：是公务员不？

      母亲：是，在税务局上班呢。

      女儿：那好，我去见见。

      这个女孩的决策过程就是典型的分类树决策。相当于通过年龄、长相、收入和是否公务员对将男人分为两个类别：见和不见。假设这个女孩对男人的要求是：30岁以下、长相中等以上并且是高收入者或中等以上收入的公务员，那么这个可以用下图表示女孩的决策逻辑（**声明：此决策树纯属为了写文章而YY的产物，没有任何根据，也不代表任何女孩的择偶倾向，请各位女同胞莫质问我^\_^**）：



      上图完整表达了这个女孩决定是否见一个约会对象的策略，其中绿色节点表示判断条件，橙色节点表示决策结果，箭头表示在一个判断条件在不同情况下的决策路径，图中红色箭头表示了上面例子中女孩的决策过程。

      这幅图基本可以算是一颗决策树，说它“基本可以算”是因为图中的判定条件没有量化，如收入高中低等等，还不能算是严格意义上的决策树，如果将所有条件量化，则就变成真正的决策树了。

      有了上面直观的认识，我们可以正式定义决策树了：

**决策树（decision tree）是一个树结构（可以是二叉树或非二叉树）。其每个非叶节点表示一个特征属性上的测试，每个分支代表这个特征属性在某个值域上的输出，而每个叶节点存放一个类别。使用决策树进行决策的过程就是从根节点开始，测试待分类项中相应的特征属性，并按照其值选择输出分支，直到到达叶子节点，将叶子节点存放的类别作为决策结果。**

      可以看到，决策树的决策过程非常直观，容易被人理解。目前决策树已经成功运用于医学、制造产业、天文学、分支生物学以及商业等诸多领域。知道了决策树的定义以及其应用方法，下面介绍决策树的构造算法。

# 3.3、决策树的构造

      不同于贝叶斯算法，决策树的构造过程不依赖领域知识，它使用属性选择度量来选择将元组最好地划分成不同的类的属性。所谓决策树的构造就是进行属性选择度量确定各个特征属性之间的拓扑结构。

      构造决策树的关键步骤是分裂属性。所谓分裂属性就是在某个节点处按照某一特征属性的不同划分构造不同的分支，其目标是让各个分裂子集尽可能地“纯”。尽可能“纯”就是尽量让一个分裂子集中待分类项属于同一类别。分裂属性分为三种不同的情况：

      1、属性是离散值且不要求生成二叉决策树。此时用属性的每一个划分作为一个分支。

      2、属性是离散值且要求生成二叉决策树。此时使用属性划分的一个子集进行测试，按照“属于此子集”和“不属于此子集”分成两个分支。

      3、属性是连续值。此时确定一个值作为分裂点split\_point，按照>split\_point和<=split\_point生成两个分支。

      构造决策树的关键性内容是进行属性选择度量，属性选择度量是一种选择分裂准则，是将给定的类标记的训练集合的数据划分D“最好”地分成个体类的[启发式方法](http://en.wikipedia.org/wiki/Heuristic_(computer_science))，它决定了拓扑结构及分裂点split\_point的选择。

      属性选择度量算法有很多，一般使用自顶向下递归分治法，并采用不回溯的[贪心策略](http://en.wikipedia.org/wiki/Greedy_algorithm)。这里介绍[ID3](http://en.wikipedia.org/wiki/ID3_algorithm)和[C4.5](http://en.wikipedia.org/wiki/C4.5_algorithm)两种常用算法。

## 3.3.1、ID3算法

      从[信息论](http://en.wikipedia.org/wiki/Information_theory)知识中我们直到，期望信息越小，[信息增益](http://en.wikipedia.org/wiki/Information_gain)越大，从而纯度越高。所以ID3算法的核心思想就是以信息增益度量属性选择，选择分裂后信息增益最大的属性进行分裂。下面先定义几个要用到的概念。

      设D为用类别对训练元组进行的划分，则D的[熵](http://en.wikipedia.org/wiki/Entropy)（entropy）表示为：

http://latex.codecogs.com/gif.latex?info(D)=-\sum%20%5em_%7bi=1%7dp_ilog_2(p_i)

      其中pi表示第i个类别在整个训练元组中出现的概率，可以用属于此类别元素的数量除以训练元组元素总数量作为估计。熵的实际意义表示是D中元组的类标号所需要的平均信息量。

      现在我们假设将训练元组D按属性A进行划分，则A对D划分的期望信息为：

http://latex.codecogs.com/gif.latex?info_A(D)=\sum%20%5ev_%7bj=1%7d\frac%7b|D_j|%7d%7b|D|%7dinfo(D_j)

      而信息增益即为两者的差值：

http://latex.codecogs.com/gif.latex?gain(A)=info(D)-info_A(D)

      ID3算法就是在每次需要分裂时，计算每个属性的增益率，然后选择增益率最大的属性进行分裂。下面我们继续用SNS社区中不真实账号检测的例子说明如何使用ID3算法构造决策树。为了简单起见，我们假设训练集合包含10个元素：



      其中s、m和l分别表示小、中和大。

      设L、F、H和R表示日志密度、好友密度、是否使用真实头像和账号是否真实，下面计算各属性的信息增益。

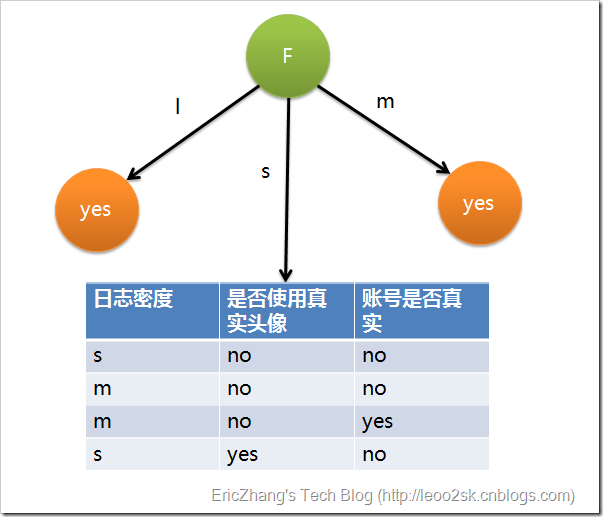
http://latex.codecogs.com/gif.latex?info(D)=-0.7log_20.7-0.3log_20.3=0.7*0.51+0.3*1.74=0.879

http://latex.codecogs.com/gif.latex?gain(L)=0.879-0.603=0.276

      因此日志密度的信息增益是0.276。

      用同样方法得到H和F的信息增益分别为0.033和0.553。

      因为F具有最大的信息增益，所以第一次分裂选择F为分裂属性，分裂后的结果如下图表示：



      在上图的基础上，再递归使用这个方法计算子节点的分裂属性，最终就可以得到整个决策树。

      上面为了简便，将特征属性离散化了，其实日志密度和好友密度都是连续的属性。对于特征属性为连续值，可以如此使用ID3算法：

      先将D中元素按照特征属性排序，则每两个相邻元素的中间点可以看做潜在分裂点，从第一个潜在分裂点开始，分裂D并计算两个集合的期望信息，具有最小期望信息的点称为这个属性的最佳分裂点，其信息期望作为此属性的信息期望。

## 3.3.2、C4.5算法

      ID3算法存在一个问题，就是偏向于多值属性，例如，如果存在唯一标识属性ID，则ID3会选择它作为分裂属性，这样虽然使得划分充分纯净，但这种划分对分类几乎毫无用处。ID3的后继算法C4.5使用[增益率](http://en.wikipedia.org/wiki/Information_gain_ratio)（gain ratio）的信息增益扩充，试图克服这个偏倚。

      C4.5算法首先定义了“分裂信息”，其定义可以表示成：

http://latex.codecogs.com/gif.latex?split\_info_A(D)=-\sum%20%5ev_%7bj=1%7d\frac%7b|D_j|%7d%7b|D|%7dlog_2(\frac%7b|D_j|%7d%7b|D|%7d)

      其中各符号意义与ID3算法相同，然后，增益率被定义为：

http://latex.codecogs.com/gif.latex?gain\_ratio(A)=\frac%7bgain(A)%7d%7bsplit\_info(A)%7d

      C4.5选择具有最大增益率的属性作为分裂属性，其具体应用与ID3类似，不再赘述。

# 3.4、关于决策树的几点补充说明

## 3.4.1、如果属性用完了怎么办

      在决策树构造过程中可能会出现这种情况：所有属性都作为分裂属性用光了，但有的子集还不是纯净集，即集合内的元素不属于同一类别。在这种情况下，由于没有更多信息可以使用了，一般对这些子集进行“[多数表决](http://en.wikipedia.org/wiki/Majority_voting)”，即使用此子集中出现次数最多的类别作为此节点类别，然后将此节点作为叶子节点。

## 3.4.2、关于剪枝

      在实际构造决策树时，通常要进行[剪枝](http://en.wikipedia.org/wiki/Pruning_(decision_trees))，这时为了处理由于数据中的噪声和离群点导致的过分拟合问题。剪枝有两种：

      先剪枝——在构造过程中，当某个节点满足剪枝条件，则直接停止此分支的构造。

      后剪枝——先构造完成完整的决策树，再通过某些条件遍历树进行剪枝

**一、病人分类的例子**

让我从一个例子开始讲起，你会看到贝叶斯分类器很好懂，一点都不难。

某个医院早上收了六个门诊病人，如下表。

　　症状　　职业　　　疾病

　　打喷嚏　护士　　　感冒   
　　打喷嚏　农夫　　　过敏   
　　头痛　　建筑工人　脑震荡   
　　头痛　　建筑工人　感冒   
　　打喷嚏　教师　　　感冒   
　　头痛　　教师　　　脑震荡

现在又来了第七个病人，是一个打喷嚏的建筑工人。请问他患上感冒的概率有多大？

根据[贝叶斯定理](http://www.ruanyifeng.com/blog/2011/08/bayesian_inference_part_one.html)：

　P(A|B) = P(B|A) P(A) / P(B)

可得

　　　P(感冒|打喷嚏x建筑工人)   
　　　　= P(打喷嚏x建筑工人|感冒) x P(感冒)   
　　　　/ P(打喷嚏x建筑工人)

假定"打喷嚏"和"建筑工人"这两个特征是独立的，因此，上面的等式就变成了

　　　P(感冒|打喷嚏x建筑工人)   
　　　　= P(打喷嚏|感冒) x P(建筑工人|感冒) x P(感冒)   
　　　　/ P(打喷嚏) x P(建筑工人)

这是可以计算的。

　　P(感冒|打喷嚏x建筑工人)   
　　　　= 0.66 x 0.33 x 0.5 / 0.5 x 0.33   
　　　　= 0.66

因此，这个打喷嚏的建筑工人，有66%的概率是得了感冒。同理，可以计算这个病人患上过敏或脑震荡的概率。比较这几个概率，就可以知道他最可能得什么病。

这就是贝叶斯分类器的基本方法：在统计资料的基础上，依据某些特征，计算各个类别的概率，从而实现分类。

**二、朴素贝叶斯分类器的公式**

假设某个体有n项特征（Feature），分别为F1、F2、...、Fn。现有m个类别（Category），分别为C1、C2、...、Cm。贝叶斯分类器就是计算出概率最大的那个分类，也就是求下面这个算式的最大值：

　P(C|F1F2...Fn)   
　　= P(F1F2...Fn|C)P(C) / P(F1F2...Fn)

由于 P(F1F2...Fn) 对于所有的类别都是相同的，可以省略，问题就变成了求

　P(F1F2...Fn|C)P(C)

的最大值。

朴素贝叶斯分类器则是更进一步，假设所有特征都彼此独立，因此

　P(F1F2...Fn|C)P(C)   
　　= P(F1|C)P(F2|C) ... P(Fn|C)P(C)

上式等号右边的每一项，都可以从统计资料中得到，由此就可以计算出每个类别对应的概率，从而找出最大概率的那个类。

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，而且有研究表明对分类结果的准确性影响不大。

下面再通过两个例子，来看如何使用朴素贝叶斯分类器。

**三、账号分类的例子**

本例摘自张洋的[《算法杂货铺----分类算法之朴素贝叶斯分类》](http://www.cnblogs.com/leoo2sk/archive/2010/09/17/1829190.html)。

根据某社区网站的抽样统计，该站10000个账号中有89%为真实账号（设为C0），11%为虚假账号（设为C1）。

　　C0 = 0.89

　　C1 = 0.11

接下来，就要用统计资料判断一个账号的真实性。假定某一个账号有以下三个特征：

　　　　F1: 日志数量/注册天数   
　　　　F2: 好友数量/注册天数   
　　　　F3: 是否使用真实头像（真实头像为1，非真实头像为0）

　　　　F1 = 0.1   
　　　　F2 = 0.2   
　　　　F3 = 0

请问该账号是真实账号还是虚假账号？

方法是使用朴素贝叶斯分类器，计算下面这个计算式的值。

　　　　P(F1|C)P(F2|C)P(F3|C)P(C)

虽然上面这些值可以从统计资料得到，但是这里有一个问题：F1和F2是连续变量，不适宜按照某个特定值计算概率。

一个技巧是将连续值变为离散值，计算区间的概率。比如将F1分解成[0, 0.05]、(0.05, 0.2)、[0.2, +∞]三个区间，然后计算每个区间的概率。在我们这个例子中，F1等于0.1，落在第二个区间，所以计算的时候，就使用第二个区间的发生概率。

根据统计资料，可得：

　　P(F1|C0) = 0.5, P(F1|C1) = 0.1   
　　P(F2|C0) = 0.7, P(F2|C1) = 0.2   
　　P(F3|C0) = 0.2, P(F3|C1) = 0.9

因此，

　　P(F1|C0) P(F2|C0) P(F3|C0) P(C0)   
　　　　= 0.5 x 0.7 x 0.2 x 0.89   
　　　　= 0.0623

　　P(F1|C1) P(F2|C1) P(F3|C1) P(C1)   
　　　　= 0.1 x 0.2 x 0.9 x 0.11   
　　　　= 0.00198

可以看到，虽然这个用户没有使用真实头像，但是他是真实账号的概率，比虚假账号高出30多倍，因此判断这个账号为真。

**四、性别分类的例子**

本例摘自[维基百科](http://en.wikipedia.org/wiki/Naive_Bayes#Sex_classification)，关于处理连续变量的另一种方法。

下面是一组人类身体特征的统计资料。

　　性别　　身高（英尺）　体重（磅）　　脚掌（英寸）

　　男 　　　6 　　　　　　180　　　　　12   
　　男 　　　5.92　　　　　190　　　　　11   
　　男 　　　5.58　　　　　170　　　　　12   
　　男 　　　5.92　　　　　165　　　　　10   
　　女 　　　5 　　　　　　100　　　　　6   
　　女 　　　5.5 　　　　　150　　　　　8   
　　女 　　　5.42　　　　　130　　　　　7   
　　女 　　　5.75　　　　　150　　　　　9

已知某人身高6英尺、体重130磅，脚掌8英寸，请问该人是男是女？

根据朴素贝叶斯分类器，计算下面这个式子的值。

P(身高|性别) x P(体重|性别) x P(脚掌|性别) x P(性别)

这里的困难在于，由于身高、体重、脚掌都是连续变量，不能采用离散变量的方法计算概率。而且由于样本太少，所以也无法分成区间计算。怎么办？

这时，可以假设男性和女性的身高、体重、脚掌都是正态分布，通过样本计算出均值和方差，也就是得到正态分布的密度函数。有了密度函数，就可以把值代入，算出某一点的密度函数的值。

比如，男性的身高是均值5.855、方差0.035的正态分布。所以，男性的身高为6英尺的概率的相对值等于1.5789（大于1并没有关系，因为这里是密度函数的值，只用来反映各个值的相对可能性）。

http://upload.wikimedia.org/math/c/c/6/cc6222c8bcc5edcdfeb67b75c4fd4c63.png

有了这些数据以后，就可以计算性别的分类了。

　　P(身高=6|男) x P(体重=130|男) x P(脚掌=8|男) x P(男)   
　　　　= 6.1984 x e-9

　　P(身高=6|女) x P(体重=130|女) x P(脚掌=8|女) x P(女)   
　　　　= 5.3778 x e-4

可以看到，女性的概率比男性要高出将近10000倍，所以判断该人为女性。

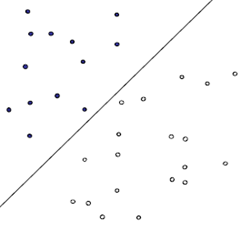
又有很长的一段时间没有更新博客了，距离上次更新已经有两个月的时间了。其中一个很大的原因是，不知道写什么好-\_-，最近一段时间看了看关于SVM(Support Vector Machine)的文章，觉得SVM是一个非常有趣，而且自成一派的方向，所以今天准备写一篇关于关于SVM的文章。

    关于SVM的论文、书籍都非常的多，引用强哥的话“SVM是让应用数学家真正得到应用的一种算法”。SVM对于大部分的普通人来说，要完全理解其中的数学是非常困难的，所以要让这些普通人理解，得要把里面的数学知识用简单的语言去讲解才行。而且想明白了这些数学，对学习其他的内容也是大有裨益的。我就是属于绝大多数的普通人，为了看明白SVM，看了不少的资料，这里把我的心得分享分享。

    其实现在能够找到的，关于SVM的中文资料已经不少了，不过个人觉得，每个人的理解都不太一样，所以还是决定写一写，一些雷同的地方肯定是不可避免的，不过还是希望能够写出一点与别人不一样的地方吧。另外本文准备不谈太多的数学（因为很多文章都谈过了），尽量简单地给出结论，就像题目一样-机器学习中的**算法**（之前叫做机器学习中的数学），**所以本系列的内容将更偏重应用一些。如果想看更详细的数学解释，可以看看参考文献中的资料。**

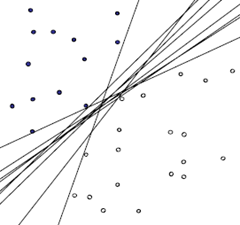
**一、线性分类器：**

**首先给出一个非常非常简单的分类问题（线性可分）**，我们要用一条直线，将下图中黑色的点和白色的点分开，很显然，图上的这条直线就是我们要求的直线之一（可以有无数条这样的直线）

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022055565831.png)    假如说，我们令黑色的点 = -1， 白色的点 =  +1，直线f(x) = w.x + b，这儿的x、w是向量，其实写成这种形式也是等价的f(x) = w1x1 + w2x2 … + wnxn + b, 当向量x的维度=2的时候，f(x) 表示二维空间中的一条直线， 当x的维度=3的时候，f(x) 表示3维空间中的一个平面，当x的维度=n > 3的时候，表示n维空间中的n-1维超平面。这些都是比较基础的内容，如果不太清楚，可能需要复习一下微积分、线性代数的内容。

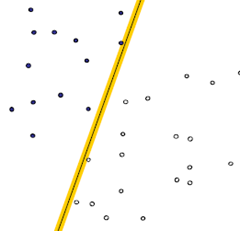
    刚刚说了，我们令黑色白色两类的点分别为+1, -1，所以当有一个新的点x需要预测属于哪个分类的时候，我们用sgn(f(x))，就可以预测了，sgn表示符号函数，当f(x) > 0的时候，sgn(f(x)) = +1, 当f(x) < 0的时候sgn(f(x)) = –1。

    但是，我们怎样才能取得一个最优的划分直线f(x)呢？下图的直线表示几条可能的f(x)

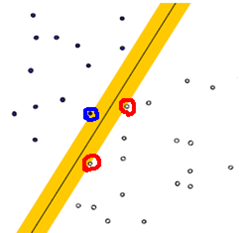
[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022055574369.png)

    一个很直观的感受是，让这条直线到给定样本中最近的点最远，这句话读起来比较拗口，下面给出几个图，来说明一下：

    第一种分法：

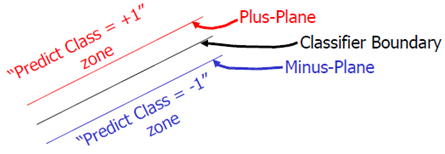
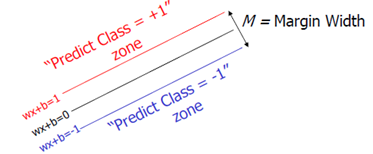
[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022055573224.png)

    第二种分法：

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022055585350.png)

    这两种分法哪种更好呢？从直观上来说，就是分割的间隙越大越好，把两个类别的点分得越开越好。就像我们平时判断一个人是男还是女，就是很难出现分错的情况，这就是男、女两个类别之间的间隙非常的大导致的，让我们可以更准确的进行分类。**在SVM中，称为Maximum Marginal，是SVM的一个理论基础之一。**选择使得间隙最大的函数作为分割平面是由很多道理的，比如说从概率的角度上来说，就是使得置信度最小的点置信度最大（听起来很拗口），从实践的角度来说，这样的效果非常好，等等。这里就不展开讲，作为一个结论就ok了，:)

    上图被红色和蓝色的线圈出来的点就是所谓的支持向量(support vector)。

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022055584204.png)    上图就是一个对之前说的类别中的间隙的一个描述。Classifier Boundary就是f(x)，红色和蓝色的线（plus plane与minus plane）就是support vector所在的面，红色、蓝色线之间的间隙就是我们要最大化的分类间的间隙。[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022055592153.png)

    这里直接给出M的式子：（从高中的解析几何就可以很容易的得到了，也可以参考后面Moore的ppt）

[image](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056009056.png)

    另外支持向量位于wx + b = 1与wx + b = -1的直线上，我们在前面乘上一个该点所属的类别y（还记得吗?y不是+1就是-1），就可以得到支持向量的表达式为：y(wx + b) = 1，这样就可以更简单的将支持向量表示出来了。

    当支持向量确定下来的时候，分割函数就确定下来了，两个问题是等价的。得到支持向量，还有一个作用是，让支持向量后方那些点就不用参与计算了。这点在后面将会更详细的讲讲。

    在这个小节的最后，给出我们要优化求解的表达式：

[image](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056002611.png)

    ||w||的意思是w的二范数，跟上面的M表达式的分母是一个意思，之前得到，M = 2 / ||w||，最大化这个式子等价于最小化||w||, 另外由于||w||是一个单调函数，我们可以对其加入平方，和前面的系数，熟悉的同学应该很容易就看出来了，这个式子是为了方便求导。

    这个式子有还有一些限制条件，完整的写下来，应该是这样的：（**原问题**）

[image](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056018433.png)

    s.t的意思是subject to，也就是在后面这个限制条件下的意思，这个词在svm的论文里面非常容易见到。这个其实是一个带约束的二次规划(quadratic programming, QP)问题，是一个凸问题，凸问题就是指的不会有局部最优解，可以想象一个漏斗，不管我们开始的时候将一个小球放在漏斗的什么位置，这个小球最终一定可以掉出漏斗，也就是得到全局最优解。s.t.后面的限制条件可以看做是一个凸多面体，我们要做的就是在这个凸多面体中找到最优解。这些问题这里不展开，因为展开的话，一本书也写不完。如果有疑问请看看wikipedia。

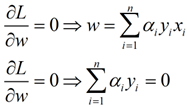
**二、转化为对偶问题，并优化求解:**

    这个优化问题可以用[拉格朗日乘子法](http://en.wikipedia.org/wiki/Lagrange_multiplier)去解，使用了[KKT条件](http://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions)的理论，这里直接作出这个式子的拉格朗日目标函数：

[image](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056015020.png)

    求解这个式子的过程需要[拉格朗日对偶性](http://en.wikipedia.org/wiki/Quadratic_programming#Lagrangian_duality)的相关知识（另外pluskid也有[一篇文章](http://blog.pluskid.org/?p=702)专门讲这个问题），并且有一定的公式推导，如果不感兴趣，**可以直接跳到后面**用**蓝色公式**表示的结论，该部分推导主要参考自[plukids的文章](http://blog.pluskid.org/?p=682)。

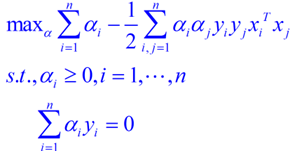
    首先让L关于w，b最小化，分别令L关于w，b的偏导数为0，得到关于**原问题**的一个表达式

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056032314.png)

    将两式带回L(w,b,a)得到对偶问题的表达式

image

    新问题加上其限制条件是（**对偶问题**）:

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056041691.png)

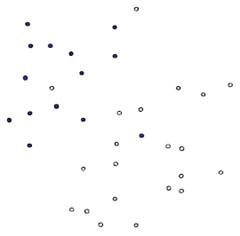
    这个就是我们需要最终优化的式子。至此，**得到了线性可分问题的优化式子**。

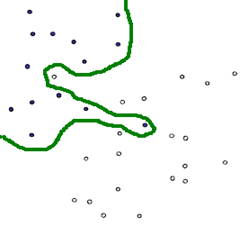
    求解这个式子，有很多的方法，比如[SMO](http://en.wikipedia.org/wiki/Sequential_minimal_optimization)等等，个人认为，求解这样的一个带约束的凸优化问题与得到这个凸优化问题是比较独立的两件事情，所以在这篇文章中准备完全不涉及如何求解这个话题，如果之后有时间可以补上一篇文章来谈谈:)。

**三、线性不可分的情况（软间隔）：**

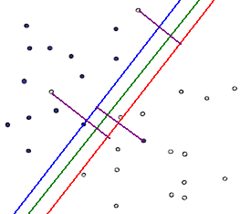
    接下来谈谈线性不可分的情况，因为**线性可分这种假设实在是太有局限性**了：

    下图就是一个典型的线性不可分的分类图，我们没有办法用一条直线去将其分成两个区域，每个区域只包含一种颜色的点。

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056043054.png)     要想在这种情况下的分类器，有两种方式，**一种是用曲线**去将其完全分开，曲线就是一种**非线性**的情况，跟之后将谈到的**核函数**有一定的关系：

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056057688.png)     **另外一种还是用直线，不过不用去保证可分性**，就是包容那些分错的情况，不过我们得加入惩罚函数，使得点分错的情况越合理越好。其实在很多时候，不是在训练的时候分类函数越完美越好，因为训练函数中有些数据本来就是噪声，可能就是在人工加上分类标签的时候加错了，如果我们在训练（学习）的时候把这些错误的点学习到了，那么模型在下次碰到这些错误情况的时候就难免出错了（假如老师给你讲课的时候，某个知识点讲错了，你还信以为真了，那么在考试的时候就难免出错）。这种学习的时候学到了“噪声”的过程就是一个过拟合（over-fitting），这在机器学习中是一个大忌，我们宁愿少学一些内容，也坚决杜绝多学一些错误的知识。还是回到主题，用直线怎么去分割线性不可分的点：

     我们可以为分错的点加上一点惩罚，对一个分错的点的**惩罚函数**就是**这个点到其正确位置的距离：**

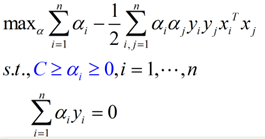
[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056057099.png)

    在上图中，蓝色、红色的直线分别为支持向量所在的边界，绿色的线为决策函数，那些紫色的线**表示分错的点到其相应的决策面的距离**，这样我们可以在原函数上面加上一个惩罚函数，并且带上其限制条件为：

image

    公式中蓝色的部分为在线性可分问题的基础上加上的惩罚函数部分，当xi在正确一边的时候，ε=0，R为全部的点的数目，C是一个由用户去指定的系数，表示对分错的点加入多少的惩罚，当C很大的时候，分错的点就会更少，但是过拟合的情况可能会比较严重，当C很小的时候，分错的点可能会很多，不过可能由此得到的模型也会不太正确，所以如何选择C是有很多学问的，不过在大部分情况下就是通过经验尝试得到的。

    接下来就是同样的，求解一个拉格朗日对偶问题，得到一个原问题的对偶问题的表达式：

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056067556.png)

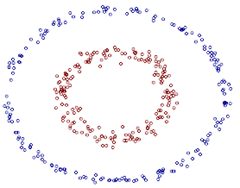
    蓝色的部分是与线性可分的对偶问题表达式的不同之处。在线性不可分情况下得到的对偶问题，不同的地方就是α的范围从[0, +∞)，变为了[0, C]，增加的惩罚ε没有为对偶问题增加什么复杂度。

**四、核函数：**

    刚刚在谈不可分的情况下，提了一句，如果使用某些非线性的方法，可以得到将两个分类完美划分的曲线，比如接下来将要说的核函数。

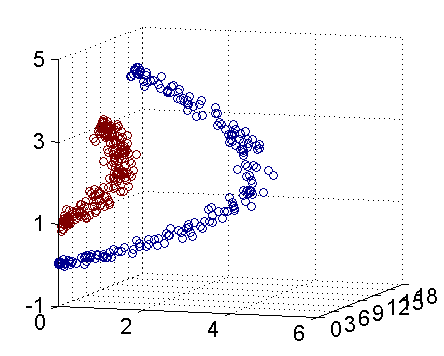
我们可以**让空间从原本的线性空间变成一个更高维的空间**，**在这个高维的线性空间下，再用一个超平面进行划分**。这儿举个例子，来理解一下如何利用空间的维度变得更高来帮助我们分类的（例子以及图片来自[pluskid的kernel函数部分](http://blog.pluskid.org/?p=685)）：

    下图是一个典型的线性不可分的情况

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056068362.png)

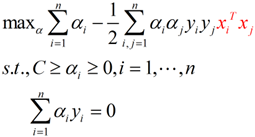
    但是当我们把这两个类似于椭圆形的点映射到一个高维空间后，映射函数为：

[image](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056071361.png)    用这个函数可以将上图的平面中的点映射到一个三维空间（z1,z2,z3)，并且对映射后的坐标加以旋转之后就可以得到一个线性可分的点集了。

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056107476.gif)

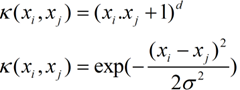
    用另外一个哲学例子来说：世界上本来没有两个完全一样的物体，对于所有的两个物体，我们可以通过增加维度来让他们最终有所区别，比如说两本书，从(颜色，内容)两个维度来说，可能是一样的，我们可以加上 **作者** 这个维度，是在不行我们还可以加入**页码**，可以加入 **拥有者**，可以加入 **购买地点**，可以加入 **笔记内容**等等。**当维度增加到无限维的时候，一定可以让任意的两个物体可分了**。

    回忆刚刚得到的对偶问题表达式：

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056126306.png)

    我们可以将红色这个部分进行改造，令：

[image](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056135717.png)     这个式子所做的事情就是将线性的空间映射到高维的空间,k(x, xj)有很多种，下面是比较典型的两种：

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201105/201105022056135128.png)    上面这个核称为多项式核，下面这个核称为高斯核，高斯核甚至是将原始空间映射为无穷维空间，另外核函数有一些比较好的性质，比如说不会比线性条件下增加多少额外的计算量，等等，这里也不再深入。一般对于一个问题，不同的核函数可能会带来不同的结果，一般是需要尝试来得到的。

**五、一些其他的问题：**

     1）如何进行多分类问题：

     上面所谈到的分类都是2分类的情况，当N分类的情况下，主要有两种方式，一种是1 vs (N – 1)一种是1 vs 1，前一种方法我们需要训练N个分类器，第i个分类器是看看是属于分类i还是属于分类i的补集（出去i的N-1个分类）。

     后一种方式我们需要训练N \* (N – 1) / 2个分类器，分类器(i,j)能够判断某个点是属于i还是属于j。

     这种处理方式不仅在SVM中会用到，在很多其他的分类中也是被广泛用到，从林教授（libsvm的作者）的结论来看，1 vs 1的方式要优于1 vs (N – 1)。

     2）SVM会overfitting吗？

     SVM避免overfitting，一种是调整之前说的惩罚函数中的C，另一种其实从式子上来看，min ||w||^2这个看起来是不是很眼熟？在最小二乘法回归的时候，我们看到过这个式子，这个式子可以让函数更平滑，所以SVM是一种不太容易over-fitting的方法。

[K-means聚类算法](http://www.cnblogs.com/jerrylead/archive/2011/04/06/2006910.html)

     K-means也是聚类算法中最简单的一种了，但是里面包含的思想却是不一般。最早我使用并实现这个算法是在学习韩爷爷那本数据挖掘的书中，那本书比较注重应用。看了Andrew Ng的这个讲义后才有些明白K-means后面包含的EM思想。

     聚类属于无监督学习，以往的回归、朴素贝叶斯、SVM等都是有类别标签y的，也就是说样例中已经给出了样例的分类。而聚类的样本中却没有给定y，只有特征x，比如假设宇宙中的星星可以表示成三维空间中的点集[clip_image002[10]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601448600.png)。聚类的目的是找到每个样本x潜在的类别y，并将同类别y的样本x放在一起。比如上面的星星，聚类后结果是一个个星团，星团里面的点相互距离比较近，星团间的星星距离就比较远了。

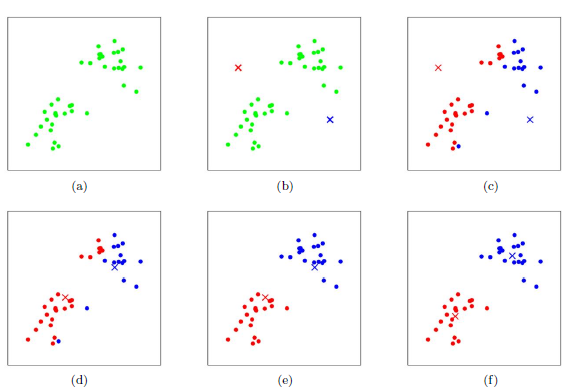
     在聚类问题中，给我们的训练样本是[clip_image004](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601448982.png)，每个[clip_image006](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601453159.png)，没有了y。

     K-means算法是将样本聚类成k个簇（cluster），具体算法描述如下：

|  |
| --- |
| 1、 随机选取k个聚类质心点（cluster centroids）为[clip_image008[6]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601454064.png)。  2、 重复下面过程直到收敛 {                 对于每一个样例i，计算其应该属于的类  [clip_image009](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601464654.png)                 对于每一个类j，重新计算该类的质心  [clip_image010[6]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601468308.png)  } |

     K是我们事先给定的聚类数，[clip_image012[6]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601473390.png)代表样例i与k个类中距离最近的那个类，[clip_image012[7]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601481537.png)的值是1到k中的一个。质心[clip_image014[6]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601496064.png)代表我们对属于同一个类的样本中心点的猜测，拿星团模型来解释就是要将所有的星星聚成k个星团，首先随机选取k个宇宙中的点（或者k个星星）作为k个星团的质心，然后第一步对于每一个星星计算其到k个质心中每一个的距离，然后选取距离最近的那个星团作为[clip_image012[8]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601504211.png)，这样经过第一步每一个星星都有了所属的星团；第二步对于每一个星团，重新计算它的质心[clip_image014[7]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601515182.png)（对里面所有的星星坐标求平均）。重复迭代第一步和第二步直到质心不变或者变化很小。

     下图展示了对n个样本点进行K-means聚类的效果，这里k取2。

[](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601529807.png)

     K-means面对的第一个问题是如何保证收敛，前面的算法中强调结束条件就是收敛，可以证明的是K-means完全可以保证收敛性。下面我们定性的描述一下收敛性，我们定义畸变函数（distortion function）如下：

[clip_image016[6]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/20110406160154496.png)

     J函数表示每个样本点到其质心的距离平方和。K-means是要将J调整到最小。假设当前J没有达到最小值，那么首先可以固定每个类的质心[clip_image014[8]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601547530.png)，调整每个样例的所属的类别[clip_image012[9]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601557629.png)来让J函数减少，同样，固定[clip_image012[10]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601569364.png)，调整每个类的质心[clip_image014[9]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601577511.png)也可以使J减小。这两个过程就是内循环中使J单调递减的过程。当J递减到最小时，[clip_image018[6]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601589562.png)和c也同时收敛。（在理论上，可以有多组不同的[clip_image018[7]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061601593042.png)和c值能够使得J取得最小值，但这种现象实际上很少见）。

     由于畸变函数J是非凸函数，意味着我们不能保证取得的最小值是全局最小值，也就是说k-means对质心初始位置的选取比较感冒，但一般情况下k-means达到的局部最优已经满足需求。但如果你怕陷入局部最优，那么可以选取不同的初始值跑多遍k-means，然后取其中最小的J对应的[clip_image018[8]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602007601.png)和c输出。

     下面累述一下K-means与EM的关系，首先回到初始问题，我们目的是将样本分成k个类，其实说白了就是求每个样例x的隐含类别y，然后利用隐含类别将x归类。由于我们事先不知道类别y，那么我们首先可以对每个样例假定一个y吧，但是怎么知道假定的对不对呢？怎么评价假定的好不好呢？我们使用样本的极大似然估计来度量，这里是就是x和y的联合分布P(x,y)了。如果找到的y能够使P(x,y)最大，那么我们找到的y就是样例x的最佳类别了，x顺手就聚类了。但是我们第一次指定的y不一定会让P(x,y)最大，而且P(x,y)还依赖于其他未知参数，当然在给定y的情况下，我们可以调整其他参数让P(x,y)最大。但是调整完参数后，我们发现有更好的y可以指定，那么我们重新指定y，然后再计算P(x,y)最大时的参数，反复迭代直至没有更好的y可以指定。

     这个过程有几个难点，第一怎么假定y？是每个样例硬指派一个y还是不同的y有不同的概率，概率如何度量。第二如何估计P(x,y)，P(x,y)还可能依赖很多其他参数，如何调整里面的参数让P(x,y)最大。这些问题在以后的篇章里回答。

     这里只是指出EM的思想，E步就是估计隐含类别y的期望值，M步调整其他参数使得在给定类别y的情况下，极大似然估计P(x,y)能够达到极大值。然后在其他参数确定的情况下，重新估计y，周而复始，直至收敛。

     上面的阐述有点费解，对应于K-means来说就是我们一开始不知道每个样例[clip_image020[10]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602017700.png)对应隐含变量也就是最佳类别[clip_image022[6]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602025847.png)。最开始可以随便指定一个[clip_image022[7]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602039534.png)给它，然后为了让P(x,y)最大（这里是要让J最小），我们求出在给定c情况下，J最小时的[clip_image014[10]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602046045.png)（前面提到的其他未知参数），然而此时发现，可以有更好的[clip_image022[8]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602059732.png)（质心与样例[clip_image020[11]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602069831.png)距离最小的类别）指定给样例[clip_image020[12]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602078851.png)，那么[clip_image022[9]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602081458.png)得到重新调整，上述过程就开始重复了，直到没有更好的[clip_image022[10]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/20110406160209685.png)指定。这样从K-means里我们可以看出它其实就是EM的体现，E步是确定隐含类别变量[clip_image024[6]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602107196.png)，M步更新其他参数[clip_image018[9]](http://images.cnblogs.com/cnblogs_com/jerrylead/201104/201104061602119247.png)来使J最小化。这里的隐含类别变量指定方法比较特殊，属于硬指定，从k个类别中硬选出一个给样例，而不是对每个类别赋予不同的概率。总体思想还是一个迭代优化过程，有目标函数，也有参数变量，只是多了个隐含变量，确定其他参数估计隐含变量，再确定隐含变量估计其他参数，直至目标函数最优。

# K-MEANS 算法

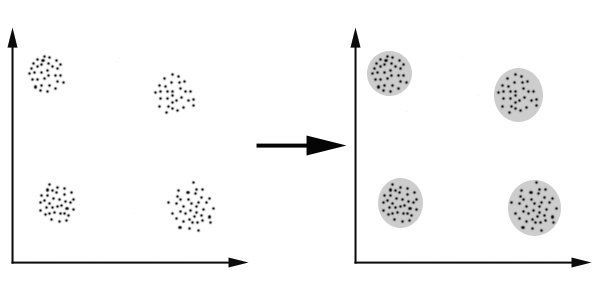
##### [2012年06月29日](http://coolshell.cn/articles/7779.html) [陈皓](http://coolshell.cn/articles/author/haoel) 评论 [74 条评论](http://coolshell.cn/articles/7779.html#comments)  67,802 人阅读

最近在学习一些数据挖掘的算法，看到了这个算法，也许这个算法对你来说很简单，但对我来说，我是一个初学者，我在网上翻看了很多资料，发现中文社区没有把这个问题讲得很全面很清楚的文章，所以，把我的学习笔记记录下来，分享给大家。

在数据挖掘中， **k-Means 算法**是一种 [cluster analysis](http://en.wikipedia.org/wiki/Cluster_analysis) 的算法，其主要是来计算数据聚集的算法，主要通过不断地取离种子点最近均值的算法。

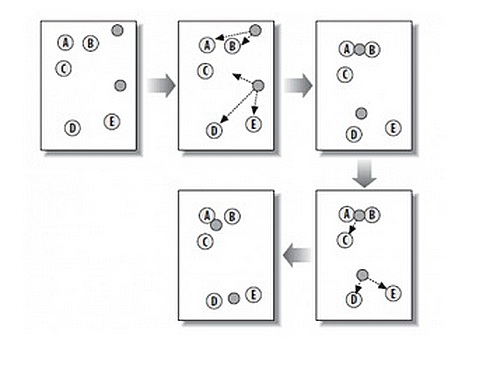
#### 问题

K-Means算法主要解决的问题如下图所示。我们可以看到，在图的左边有一些点，我们用肉眼可以看出来有四个点群，但是我们怎么通过计算机程序找出这几个点群来呢？于是就出现了我们的K-Means算法（[Wikipedia链接](http://en.wikipedia.org/wiki/K-means_clustering)）

K-Means 要解决的问题

#### 算法概要

这个算法其实很简单，如下图所示：

K-Means 算法概要

从上图中，我们可以看到，**A, B, C, D, E 是五个在图中点。而灰色的点是我们的种子点，也就是我们用来找点群的点**。有两个种子点，所以K=2。

然后，K-Means的算法如下：

1. 随机在图中取K（这里K=2）个种子点。
2. 然后对图中的所有点求到这K个种子点的距离，假如点Pi离种子点Si最近，那么Pi属于Si点群。（上图中，我们可以看到A,B属于上面的种子点，C,D,E属于下面中部的种子点）
3. 接下来，我们要移动种子点到属于他的“点群”的中心。（见图上的第三步）
4. 然后重复第2）和第3）步，直到，种子点没有移动（我们可以看到图中的第四步上面的种子点聚合了A,B,C，下面的种子点聚合了D，E）。

这个算法很简单，但是有些细节我要提一下，求距离的公式我不说了，大家有初中毕业水平的人都应该知道怎么算的。我重点想说一下“求点群中心的算法”

#### 求点群中心的算法

一般来说，求点群中心点的算法你可以很简的使用各个点的X/Y坐标的平均值。不过，我这里想告诉大家另三个求中心点的的公式：

**1）Minkowski Distance 公式 ——** λ 可以随意取值，可以是负数，也可以是正数，或是无穷大。

http://coolshell.cn/wp-content/uploads/2012/06/MinkowskiDistance_clip_image102.gif

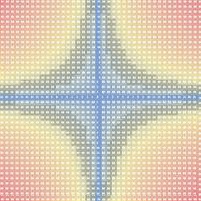
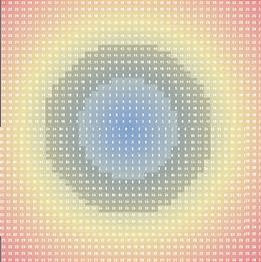
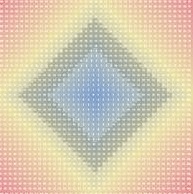
**2）Euclidean Distance 公式**—— 也就是第一个公式 λ=2 的情况

http://coolshell.cn/wp-content/uploads/2012/06/EuclideanDistance_clip_image002.gif

**3）CityBlock Distance 公式**—— 也就是第一个公式 λ=1 的情况

http://coolshell.cn/wp-content/uploads/2012/06/CityBlockDistance_clip_image002.gif

这三个公式的求中心点有一些不一样的地方，我们看下图（对于第一个 λ 在 0-1之间）。

**（1）Minkowski Distance     （2）Euclidean Distance    （3） CityBlock Distance**

上面这几个图的大意是他们是怎么个逼近中心的，第一个图以星形的方式，第二个图以同心圆的方式，第三个图以菱形的方式。

#### K-Means的演示

如果你以”[K Means Demo](https://www.google.com/search?hl=zh-CN&q=K+Means+Demo)“为关键字到Google里查你可以查到很多演示。这里推荐一个演示

<http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html>

操作是，鼠标左键是初始化点，右键初始化“种子点”，然后勾选“Show History”可以看到一步一步的迭代。

注：这个演示的链接也有一个不错的 [K Means Tutorial](http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/index.html) 。

#### K-Means ++ 算法

K-Means主要有两个最重大的缺陷——都和初始值有关：

* K 是事先给定的，这个 K 值的选定是非常难以估计的。很多时候，事先并不知道给定的数据集应该分成多少个类别才最合适。（ [ISODATA 算法](http://en.wikipedia.org/wiki/Multispectral_pattern_recognition)通过类的自动合并和分裂，得到较为合理的类型数目 K）
* K-Means算法需要用初始随机种子点来搞，这个随机种子点太重要，不同的随机种子点会有得到完全不同的结果。（[K-Means++算法](http://en.wikipedia.org/wiki/K-means%2B%2B)可以用来解决这个问题，其可以有效地选择初始点）

我在这里重点说一下 K-Means++算法步骤：

1. 先从我们的数据库随机挑个随机点当“种子点”。
2. 对于每个点，我们都计算其和最近的一个“种子点”的距离D(x)并保存在一个数组里，然后把这些距离加起来得到Sum(D(x))。
3. 然后，再取一个随机值，用权重的方式来取计算下一个“种子点”。这个算法的实现是，先取一个能落在Sum(D(x))中的随机值Random，然后用Random -= D(x)，直到其<=0，此时的点就是下一个“种子点”。
4. 重复第（2）和第（3）步直到所有的K个种子点都被选出来。
5. 进行K-Means算法。

相关的代码你可以在这里找到“[implement the K-means++ algorithm](http://rosettacode.org/wiki/K-means%2B%2B_clustering)”(墙) 另，[Apache 的通用数据学库也实现了这一算法](http://commons.apache.org/math/api-2.1/index.html?org/apache/commons/math/stat/clustering/KMeansPlusPlusClusterer.html)

#### K-Means 算法应用

看到这里，你会说，K-Means算法看来很简单，而且好像就是在玩坐标点，没什么真实用处。而且，这个算法缺陷很多，还不如人工呢。是的，前面的例子只是玩二维坐标点，的确没什么意思。但是你想一下下面的几个问题：

1）如果不是二维的，是多维的，如5维的，那么，就只能用计算机来计算了。

2）二维坐标点的X, Y 坐标，其实是一种向量，是一种数学抽象。现实世界中很多属性是可以抽象成向量的，比如，我们的年龄，我们的喜好，我们的商品，等等，能抽象成向量的目的就是可以让计算机知道某两个属性间的距离。如：我们认为，18岁的人离24岁的人的距离要比离12岁的距离要近，鞋子这个商品离衣服这个商品的距离要比电脑要近，等等。

**只要能把现实世界的物体的属性抽象成向量，就可以用K-Means算法来归类了**。

在 《[k均值聚类(K-means)](http://www.cnblogs.com/leoo2sk/archive/2010/09/20/k-means.html)》 这篇文章中举了一个很不错的应用例子，作者用亚洲15支足球队的2005年到1010年的战绩做了一个向量表，然后用K-Means把球队归类，得出了下面的结果，呵呵。

* 亚洲一流：日本，韩国，伊朗，沙特
* 亚洲二流：乌兹别克斯坦，巴林，朝鲜
* 亚洲三流：中国，伊拉克，卡塔尔，阿联酋，泰国，越南，阿曼，印尼

其实，这样的业务例子还有很多，比如，分析一个公司的客户分类，这样可以对不同的客户使用不同的商业策略，或是电子商务中分析商品相似度，归类商品，从而可以使用一些不同的销售策略，等等。

# K NEAREST NEIGHBOR 算法

##### [2012年08月17日](http://coolshell.cn/articles/8052.html) [陈皓](http://coolshell.cn/articles/author/haoel) 评论 [45 条评论](http://coolshell.cn/articles/8052.html#comments)  32,830 人阅读

K Nearest Neighbor算法又叫KNN算法，这个算法是机器学习里面一个比较经典的算法， 总体来说KNN算法是相对比较容易理解的算法。其中的K表示最接近自己的K个数据样本。KNN算法和[K-Means算法](http://coolshell.cn/articles/7779.html)不同的是，K-Means算法用来聚类，用来判断哪些东西是一个比较相近的类型，而KNN算法是用来做归类的，也就是说，有一个样本空间里的样本分成很几个类型，然后，给定一个待分类的数据，通过计算接近自己最近的K个样本来判断这个待分类数据属于哪个分类。**你可以简单的理解为由那离自己最近的K个点来投票决定待分类数据归为哪一类**。

Wikipedia上的[KNN词条](http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm)中有一个比较经典的图如下：



从上图中我们可以看到，图中的有两个类型的样本数据，一类是蓝色的正方形，另一类是红色的三角形。而那个绿色的圆形是我们待分类的数据。

* 如果K=3，那么离绿色点最近的有2个红色三角形和1个蓝色的正方形，这3个点投票，于是绿色的这个待分类点属于红色的三角形。
* 如果K=5，那么离绿色点最近的有2个红色三角形和3个蓝色的正方形，这5个点投票，于是绿色的这个待分类点属于蓝色的正方形。

我们可以看到，机器学习的本质——**是基于一种数据统计的方法**！那么，这个算法有什么用呢？我们来看几个示例。

#### 产品质量判断

假设我们需要判断纸巾的品质好坏，纸巾的品质好坏可以抽像出两个向量，一个是“酸腐蚀的时间”，一个是“能承受的压强”。如果我们的样本空间如下：（所谓样本空间，又叫Training Data，也就是用于机器学习的数据）

|  |  |  |
| --- | --- | --- |
| **向量X1**  **耐酸时间（秒）** | **向量X2**  **圧强(公斤/平方米)** | **品质Y** |
| 7 | 7 | 坏 |
| 7 | 4 | 坏 |
| 3 | 4 | 好 |
| 1 | 4 | 好 |

那么，如果 X1 = 3 和 X2 = 7， 这个毛巾的品质是什么呢？这里就可以用到KNN算法来判断了。

假设K=3，K应该是一个奇数，这样可以保证不会有平票，下面是我们计算（3，7）到所有点的距离。（关于那些距离公式，可以参看[K-Means算法中的距离公式](http://coolshell.cn/articles/7779.html)）

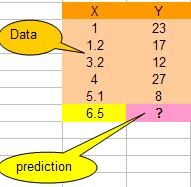
|  |  |  |  |
| --- | --- | --- | --- |
| **向量X1**  **耐酸时间（秒）** | **向量X2**  **圧强(公斤/平方米)** | **计算到 (3, 7)的距离** | **向量Y** |
| 7 | 7 | **http://coolshell.cn/wp-content/uploads/2012/08/KNN_Numerical-example_clip_image004.gif** | 坏 |
| 7 | 4 | **http://coolshell.cn/wp-content/uploads/2012/08/KNN_Numerical-example_clip_image006.gif** | N/A |
| 3 | 4 | **http://coolshell.cn/wp-content/uploads/2012/08/KNN_Numerical-example_clip_image008.gif** | 好 |
| 1 | 4 | **http://coolshell.cn/wp-content/uploads/2012/08/KNN_Numerical-example_clip_image010.gif** | 好 |

所以，最后的投票，好的有2票，坏的有1票，最终需要测试的（3，7）是合格品。（当然，你还可以使用权重——可以把距离值做为权重，越近的权重越大，这样可能会更准确一些）

**注：**[**示例来自这里**](http://people.revoledu.com/kardi/tutorial/KNN/KNN_Numerical-example.html)**，**[**K-NearestNeighbors Excel表格下载**](http://coolshell.cn/wp-content/uploads/2012/08/K-NearestNeighbors.xls)

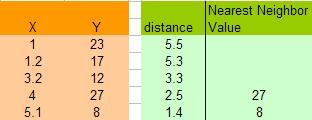
#### 预测

假设我们有下面一组数据，假设X是流逝的秒数，Y值是随时间变换的一个数值（你可以想像是股票值）



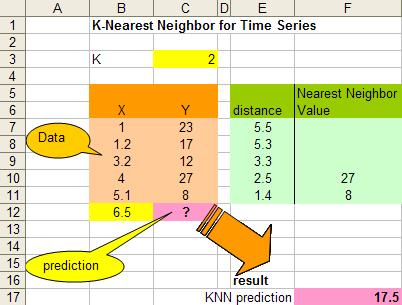
那么，当时间是6.5秒的时候，Y值会是多少呢？我们可以用KNN算法来预测之。

这里，让我们假设K=2，于是我们可以计算所有X点到6.5的距离，如：X=5.1，距离是 | 6.5 – 5.1 | = 1.4， X = 1.2 那么距离是 | 6.5 – 1.2 | = 5.3 。于是我们得到下面的表：



注意，上图中因为K=2，所以得到X=4 和 X =5.1的点最近，得到的Y的值分别为27和8，在这种情况下，我们可以简单的使用平均值来计算：http://coolshell.cn/wp-content/uploads/2012/08/KNN_TimeSeries_clip_image008.gif

于是，最终预测的数值为：17.5



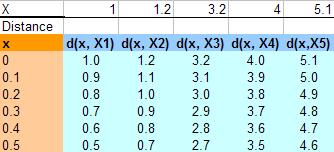
**注：**[**示例来自这里**](http://people.revoledu.com/kardi/tutorial/KNN/KNN_TimeSeries.htm)**，**[**KNN\_TimeSeries Excel表格下载**](http://coolshell.cn/wp-content/uploads/2012/08/KNN_TimeSeries.xls)

#### 插值，平滑曲线

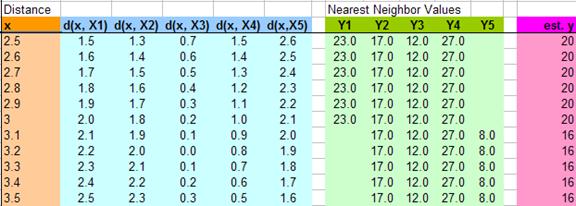
KNN算法还可以用来做平滑曲线用，这个用法比较另类。假如我们的样本数据如下（和上面的一样）：

http://coolshell.cn/wp-content/uploads/2012/08/KNN_TimeSeries_clip_image012.jpg

要平滑这些点，我们需要在其中插入一些值，比如我们用步长为0.1开始插值，从0到6开始，计算到所有X点的距离（绝对值），下图给出了从0到0.5 的数据：



下图给出了从2.5到3.5插入的11个值，然后计算他们到各个X的距离，假值K=4，那么我们就用最近4个X的Y值，然后求平均值，得到下面的表：



于是可以从0.0, 0.1, 0.2, 0.3 …. 1.1, 1.2, 1.3…..3.1, 3.2…..5.8, 5.9, 6.0 一个大表，跟据K的取值不同，得到下面的图：

|  |  |
| --- | --- |
| http://coolshell.cn/wp-content/uploads/2012/08/KNN_TimeSeries_clip_image026.jpg | http://coolshell.cn/wp-content/uploads/2012/08/KNN_TimeSeries_clip_image024.jpg |
| http://coolshell.cn/wp-content/uploads/2012/08/KNN_TimeSeries_clip_image022.jpg | http://coolshell.cn/wp-content/uploads/2012/08/KNN_TimeSeries_clip_image020.jpg |
| http://coolshell.cn/wp-content/uploads/2012/08/KNN_TimeSeries_clip_image018.jpg |  |

**注：**[**示例来自这里**](http://people.revoledu.com/kardi/tutorial/KNN/KNN_TimeSeries.htm)**，**[**KNN\_Smoothing Excel表格下载**](http://coolshell.cn/wp-content/uploads/2012/08/KNN_Smoothing.xls)

#### 后记

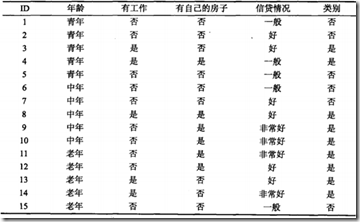
最后，我想再多说两个事，

1） 一个是机器学习，算法基本上都比较简单，最难的是数学建模，把那些业务中的特性抽象成向量的过程，另一个是选取适合模型的数据样本。这两个事都不是简单的事。算法反而是比较简单的事。

2）对于KNN算法中找到离自己最近的K个点，是一个很经典的算法面试题，需要使用到的数据结构是“[最大堆——Max Heap](http://en.wikipedia.org/wiki/Binary_heap)”，一种二叉树。你可以看看相关的算法。

# 1、决策树基本问题

## 1.1 ****定义****

**[](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929222610949-2025142699.png)**

**我们应该设计什么的**[**算法**](http://lib.csdn.net/base/datastructure)**，使得计算机对贷款申请人员的申请信息自动进行分类，以决定能否贷款?**

**一个女孩的母亲要给这个女孩介绍男朋友，于是有了下面的对话：**

**女儿：多大年纪了？**

**母亲：26。**

**女儿：长的帅不帅？**

**母亲：挺帅的。**

**女儿：收入高不？**

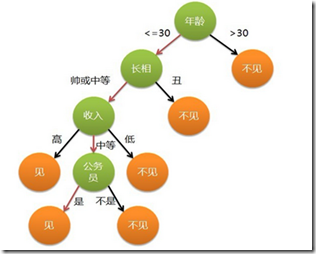
**母亲：不算很高，中等情况。**

**女儿：是公务员不？**

**母亲：是，在税务局上班呢。**

**女儿：那好，我去见见。**

**决策过程：**

**[](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929222630386-179635683.png)**

**这个女孩的决策过程就是典型的分类树决策。相当于通过年龄、长相、收入和是否公务员对将男人分为两个类别：见和不见**

**决策树是一种描述对样本实例(男人)进行分类(见或不见)的树形结构。**

**决策树由结点和有向边组成。最上部是根节点，此时所有样本都在一起，经过该节点后样本被划分到各子节点中。每个子节点再用新的特征来进一步决策，直到最后的叶节点。叶节点上只包含单纯一类样本(见或不见)，不需要在进行划分。**

**结点两种类型:内部结点和叶结点。**

**内部结点表示一个特征或属性，叶节点表示一个类。**

## 1.2 熵

**首先，我们该选择什么标准(属性、特征)作为我们的首要条件(根节点)对样本(男人)进行划分，决定见或不见呢？——特征选择**

**母亲希望女儿能最快速的有一个明确的态度，决定见或不见，这样好给男方一个明确的答复。**

**母亲需要获得尽可能多的信息，减少不确定性。**

**信息的如何度量？——熵**

**母亲得到信息越多，女儿的态度越明确，与男方见与不见的不确定性越低。因此，信息量与不确定性相对应。使用熵来表示不确定性的度量。**

**熵定义：如果一件事有k种可的结果，每种结果的概率为**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223432418-674022572.png)**

**则我们对此事件的结果进行观察后得到的信息量为:**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223434136-458414320.png)**

**熵越大，随机变量(见与不见)的不确定性越大。**

## 1.3 条件熵(局部，现象发生的前提下的熵)

**条件熵H(Y|X)表示在已知随机变量X的条件下随机变量Y的不确定性。例如，知道男生年龄的前提条件下，根据女儿见与不见的不确定性。**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223435511-1862652845.png)**

**熵与条件熵中概率由数据估计得到时，所对应的熵和条件熵称为经验熵和经验条件熵。若概率为0，令0log0=0**

## 1.4 ****信息增益****

**信息增益表示得知特征X(年龄)的信息使得类Y(见与不见)的信息的不确定性减少程度。**

**特征A对训练数据集D的信息增益g(D,A)，定义为集合D的经验熵H(D)与特征A给定条件下的经验条件熵H(D|A)之差**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223440746-1947706262.png)**

**熵H(Y)与条件熵H(Y|X)之差称为互信息，即g(D,A)。**

**信息增益大表明信息增多，信息增多，则不确定性就越小，母亲应该选择使得信息增益增大的条件询问女儿。**

1.5 信息增益准则的特征选择方法

**对数据集D，计算每个特征的信息增益，并比较他们的大小，选择信息增益最大的特征。**

## ****1.6 信息增益率****

**信息增益率定义:特征A对训练数据集D的信息增益比定义为其信息增益与训练数据D关于特征A的值的熵HA(D)之比**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929224325949-1778696163.png)**

**其中，[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929224329090-143344929.png) ，n是特征A取值个数。如A代表年龄。**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929224330949-1106305054.png)**

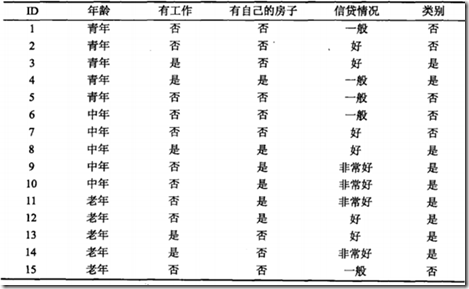
# ****2 ID3****

## 2.1 ID3 的定义

**ID3算法的核心是在决策树各个子节点上应用信息增益准则选择特征，递归的构建决策树，具体方法是:从根节点开始，对节点计算所有可能的特征的信息增益，选择信息增益最大的特征作为节点的特征，由该特征的不同取值建立子节点；再对子节点递归调用以上方法，构建决策树。**

**直到所有特征的信息增益均很小或没有特征可以选择为止。最后得到一个决策树。**

**例子：贷款申请样本数据表**

**[](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223452230-383300489.png)**

**根据贷款申请样本数据表，我们有15条样本记录，则样本容量为15。最终分为是否贷款2个类，其中是有9条记录，否有6条记录。有年龄、有工作、有自己的房子和信贷情况4个不同特征。每个特征有不同的取值，如年龄有老、中、青3种取值。**

**由熵的定义：**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223457480-337695179.png)**

**计算经验熵：**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223459168-1412161017.png)**

**然后计算各特征对数据集D的信息增益。分别以A1，A2，A3，A4表示年龄、有工作、有自己的房子和信贷情况4个特征。**

**根据年龄有取值青年、中年、老年。**

**青年贷款是2条记录，否3条记录，共5条记录**

**中年贷款是3条记录，否2条记录，共5条记录**

**老年贷款是4条记录，否1条记录，共5条记录**

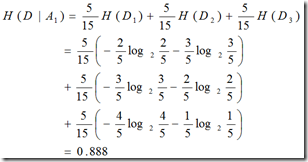
**由条件熵公式**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223503465-1761033989.png)**

**条件熵公式**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223507918-1993900184.png)**

**年龄为已知条件的条件熵为**

**[](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223515855-1820762209.png)**

**D1，D2，D3分别是年龄取值为青年、中年、老年的样本子集。**

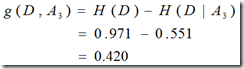
**以年龄为条件的信息增益为**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223517902-1965284942.png)**

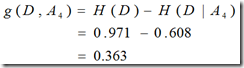
**有工作的信息增益**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223519121-286512891.png)**

**有房子的信息增益**

**[](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223520433-623064686.png)**

**信贷情况的信息增益**

**[](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929223522793-874593292.png)**

**最后比较各特征的信息增益值，对于特征A3有自己房子的信息增益值最大，所以选择特征A3作为最优特征。**

**由于特征A3(有自己房子)的信息增益值最大，所以选择特征A3作为根节点的特征。它将训练数据集划分为两个子集D1(A3取值为是)和D2(A3取值为否)。由于D1只有同一类样本点，可以明确要贷款给D1，所以它成为一个叶节点，节点类标记为“是”。**

**对于D2则需要从特征A1(年龄)，A2(有工作)和A4(信贷情况)中选择新的特征。计算各个特征的信息增益:**

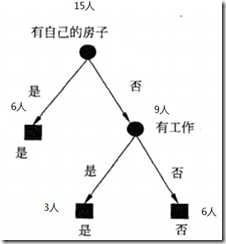
**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929224310043-1221495953.png)**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929224317652-1263132473.png)**

**[image](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929224319136-1285455101.png)**

**选择信息增益最大的特征A2(有工作)作为节点特征。A2有2个取值，一个对应“是”(有工作)的子节点，包含3个样本，他们属于同一类，所以这是一个叶节点，类标记为“是”；另一个对应“否”(无工作)的子节点，包含6个样本，属于同一类，这也是一个叶节点，类标记为“否”。**

**换句话有15个贷款人，经过是否有房这一筛选条件，有房子的6个人能够贷款。剩余9个人需要进一步筛选，以是否有工作为筛选条件，有工作的3个人可以贷款，无工作的6个人不能够贷款。**

**[](http://images2015.cnblogs.com/blog/456673/201509/456673-20150929224323386-586911.png)**

**该决策树只用了两个特征(有两个内部结点)，以有自己的房子作为首要判决条件，然后以有工作作为判决条件是否可以贷款。**

**ID3算法只有树的生成，所以该算法生成的树容易产生过拟合，分得太细，考虑条件太多。**

## 2.2 ID3 的缺点

**1.用信息增益选择属性时偏向于选择分枝比较多的属性值，即取值 多的属性。**

**2.不能处理连续属性。**

# 3、C4.5

**C4.5算法是数据挖掘十大算法之一，它是对ID3算法的改进，相对于ID3算法主要有以下几个改进**

**（1）用信息增益比来选择属性**

**（2）在决策树的构造过程中对树进行剪枝**

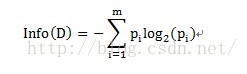
**（3）对非离散数据也能处理**

**（4）能够对不完整数据进行处理**

以下例子以ID3的过程为主，穿插着增添了C4.5的特性：

本文采用评价电信服务保障中的满意度预警专题来解释决策树算法，即假如我家办了电信的宽带，有一天宽带不能上网了，于是我打电话给电信报修，然后电信派相关人员进行维修，修好以后电信的回访专员询问我对这次修理障碍的过程是否满意，我会给我对这次修理障碍给出相应评价，满意或者不满意。根据历史数据可以建立满意度预警模型，建模的目的就是为了预测哪些用户会给出不满意的评价。目标变量为二分类变量：满意（记为0）和不满意（记为1）。自变量为根据修理障碍过程产生的数据，如障碍类型、障碍原因、修障总时长、最近一个月发生故障的次数、最近一个月不满意次数等等。简单的数据如下：  
客户ID    故障原因    故障类型    修障时长       满意度  
001    1    5   10.2   1  
002    1    5   12   0  
003  1    5    14   1  
004    2     5    16  0  
005   2    5   18  1  
006   2    6    20   0  
007    3     6   22 1  
008    3    6   23   0  
009    3    6   24    1  
010    3    6   25    0  
故障原因和故障类型都为离散型变量，数字代表原因ID和类型ID。修障时长为连续型变量，单位为小时。满意度中1为不满意、0为满意。  
    下面沿着分裂属性的选择和树剪枝两条主线，去描述三种决策树算法构造满意度预警模型：  
    分裂属性的选择：即该选择故障原因、故障类型、修障时长三个变量中的哪个作为决策树的第一个分支。  
ID3算法是采用信息增益来选择树叉，c4.5算法采用增益率，CART算法采用Gini指标。此外离散型变量和连续型变量在计算信息增益、增益率、Gini指标时会有些区别。详细描述如下：  
    1.ID3算法的信息增益：  
     信息增益的思想来源于信息论的香农定理，ID3算法选择具有最高信息增益的自变量作为当前的树叉（树的分支），以满意度预警模型为例，模型有三个自变量：故障原因、故障类型、修障时长。分别计算三个自变量的信息增益，选取其中最大的信息增益作为树叉。信息增益=原信息需求-要按某个自变量划分所需要的信息。  
如以自变量故障原因举例，故障原因的信息增益=原信息需求（即仅仅基于满意度类别比例的信息需求，记为a）-按照故障原因划分所需要的信息需求（记为a1）。

其中原信息需求a的计算方式为:

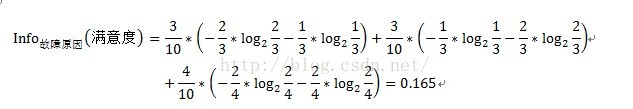


其中D为目标变量，此例中为满意度。m=2，即满意和不满意两种情况。Pi为满意度中属于分别属于满意和不满意的概率。此例中共计10条数据，满意5条，不满意5条。概率都为1/2。Info（满意度）即为仅仅基于满意和满意的类别比例进行划分所需要的信息需求，计算方式为：

http://img.blog.csdn.net/20150811095941434?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

按照故障原因划分所需要的信息需求（记为a1）可以表示为：

http://img.blog.csdn.net/20150811095952634?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

其中A表示目标变量D（即满意度）中按自变量A划分所需要的信息，即按故障类型进行划分所需要的信息。V表示在目标变量D（即满意度）中，按照自变量A（此处为故障原因）进行划分，即故障原因分别为1、2、3进行划分，将目标变量分别划分为3个子集，{D1、D2、D3}，因此V=3。即故障原因为1的划分中，有2个不满意和1个满意。D1即指2个不满意和1个满意。故障原因为2的划分中，有1个不满意和2个满意。D2即指1个不满意和2个满意。故障原因为3的划分中，有2个不满意和2个满意。D3即指2个不满意和2个满意。具体公式如下：  
  
  
注：此处的计算结果即0.165不准确，没有真正去算，结果仅供参考。  
因此变量故障原因的信息增益Gain(故障原因)=Info(满意度)- Info故障原因（满意度）=1-0.165=0.835

同样的道理，变量故障类型的信息增益计算方式如下：

http://img.blog.csdn.net/20150811100034333?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

http://img.blog.csdn.net/20150811100040129?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

                    =0.205（结果不准，为准确计算）

变量故障类型的信息增益Gain(故障类型)=1-0.205=0.795

故障原因和故障类型两个变量都是离散型变量，按上述方式即可求得信息增益，但修障时长为连续型变量，对于连续型变量该怎样计算信息增益呢？

（此处的方法来自于C4.5）

只需将连续型变量由小到大递增排序，取相邻两个值的中点作为分裂点，然后按照离散型变量计算信息增益的方法计算信息增益，取其中最大的信息增益作为最终的分裂点。如求修障时长的信息增益，首先将修障时长递增排序，即10.2、12、14、16、18、20、22、23、24、25,取相邻两个值的中点，如10.2和12，中点即为（10.2+12）/2=11.1,同理可得其他中点，分别为11.1、13、15、17、19、21、22.5、23.5、24.5。对每个中点都离散化成两个子集，如中点11.1，可以离散化为两个<=11.1和>11.1两个子集，然后按照离散型变量的信息增益计算方式计算其信息增益，如中点11.1的信息增益计算过程如下：

http://img.blog.csdn.net/20150811100102247?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Centerhttp://img.blog.csdn.net/20150811100108982?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

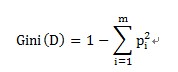
中点11.1的信息增益Gain(修障时长)=1-0.222=0.778  
中点13的信息增益计算过程如下：  
http://img.blog.csdn.net/20150811100116964?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Centerhttp://img.blog.csdn.net/20150811100122943?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center  
中点11.1的信息增益Gain(修障时长)=1-1=0  
同理分别求得各个中点的信息增益，选取其中最大的信息增益作为分裂点，如取中点11.1。然后与故障原因和故障类型的信息增益相比较，取最大的信息增益作为第一个树叉的分支，此例中选取了故障原因作为第一个分叉。按照同样的方式继续构造树的分支。  
    总之，信息增益的直观解释为选取按某个自变量划分所需要的期望信息，该期望信息越小，划分的纯度越高。因为对于某个分类问题而言，Info(D)都是固定的，而信息增益Gain(A)=Info(D)-InfoA(D)  影响信息增益的关键因素为：-InfoA(D)，即按自变量A进行划分，所需要的期望信息越小，整体的信息增益越大，越能将分类变量区分出来。

由于信息增益选择分裂属性的方式会倾向于选择具有大量值的属性（即自变量），如对于客户ID，每个客户ID对应一个满意度，即按此变量划分每个划分都是纯的（即完全的划分，只有属于一个类别），客户ID的信息增益为最大值1。但这种按该自变量的每个值进行分类的方式是没有任何意义的。为了克服这一弊端，有人提出了采用增益率（GainRate）来选择分裂属性。计算方式如下：  
http://img.blog.csdn.net/20150811100130987?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Centerhttp://img.blog.csdn.net/20150811100139415?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center  
  
其中Gain(A)的计算方式与ID3算法中的信息增益计算方式相同。  
以故障原因为例：  
http://img.blog.csdn.net/20150811100144374?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center  
                         =1.201  
  Gain(故障原因)=0.835（前文已求得）  
    GainRate故障原因（满意度）=1.201/0.835=1.438  
同理可以求得其他自变量的增益率。  
选取最大的信息增益率作为分裂属性。

# 4、CART

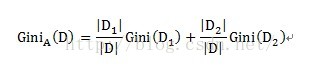
 CART算法选择分裂属性的方式是比较有意思的，首先计算不纯度，然后利用不纯度计算Gini指标。以满意度预警模型为例，计算自变量故障原因的Gini指标时，先按照故障原因可能的子集进行划分，即可以将故障原因具体划分为如下的子集：{1,2,3}、{1，2}、{1,3}、{2,3}、{1}、{2}、{3}、{}，共计8（2^V）个子集。由于{1,2,3}和{}对于分类来说没有任何意义，因此实际分为2^V-2共计6个有效子集。然后计算这6个有效子集的不纯度和Gini指标，选取最小的Gini指标作为分裂属性。

不纯度的计算方式为：



pi表示按某个变量划分中，目标变量不同类别的概率。

某个自变量的Gini指标的计算方式如下：



        对应到满意度模型中，A为自变量，即故障原因、故障类型、修障时长。D代表满意度，D1和D2分别为按变量A的子集所划分出的两个不同元组，如按子集{1,2}划分，D1即为故障原因属于{1,2}的满意度评价，共有6条数据，D2即故障原因不属于{1,2}的满意度评价，共有3条数据。计算子集{1,2}的不纯度时，即Gini（D1），在故障原因属于{1,2}的样本数据中，分别有3条不满意和3条满意的数据，因此不纯度为1-(3/6)^2-(3/6)^2=0.5。  
        以故障原因为例，计算过程如下：

http://img.blog.csdn.net/20150811100157365?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

http://img.blog.csdn.net/20150811100203560?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

                                                          =0.5  
计算子集故障原因={1,3}的子集的Gini指标时，D1和D2分别为故障原因={1,3}的元组共计7条数据，故障原因不属于{1,3}的元组即故障原因为2的数据，共计3条数据。详细计算过程如下：

http://img.blog.csdn.net/20150811100209124?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

http://img.blog.csdn.net/20150811100214016?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center  
=0.52

同理可以计算出故障原因的每个子集的Gini指标，按同样的方式还可以计算故障类型和修障时长每个子集的Gini指标，选取其中最小的Gini指标作为树的分支（ Gini（D）越小，则数据集D的纯度越高）。连续型变量的离散方式与信息增益中的离散方式相同。

# ****6、随机森林****

在[**机器学习**](http://lib.csdn.net/base/2)中，随机森林由许多的决策树组成，因为这些决策树的形成采用了随机的方法，因此也叫做随机决策树。随机森林中的树之间是没有关联的。当测试数据进入随机森林时，其实就是让每一颗决策树进行分类，最后取所有决策树中分类结果最多的那类为最终的结果。因此随机森林是一个包含多个决策树的分类器，并且其输出的类别是由个别树输出的类别的众数而定。随机森林可以既可以处理属性为离散值的量，如ID3算法，也可以处理属性为连续值的量，比如C4.5算法。另外，随机森林还可以用来进行无监督学习聚类和异常点检测。

## 6.1 理论描述

随机森林由决策树组成，决策树实际上是将空间用超平面进行划分的一种方法，每次分割的时候，都将当前的空间一分为二，如说下面的决策树，其属性的值都是连续的实数，如图1所示。将空间划分为成的样子如图2所示(注：所使用图片来自于网络)。

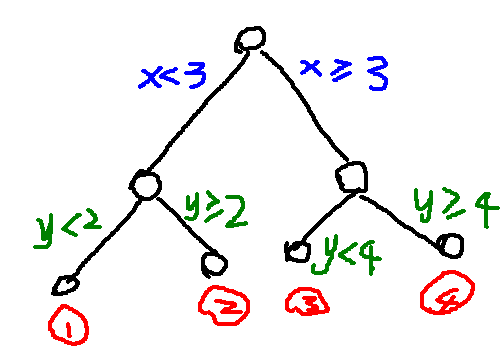
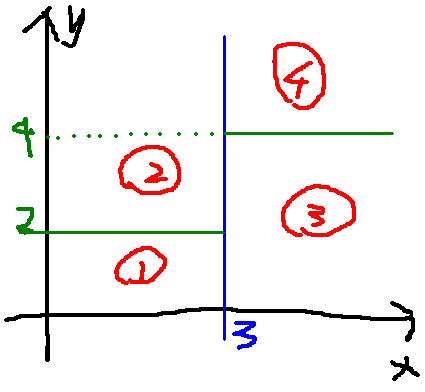
 

图1                                                                  图2

随机森林比较适合做多分类问题，训练和预测速度快；同时，对训练数据的容错能力，是一种有效地估计缺失数据的一种方法，当数据集中有大比例的数据缺失时仍然可以保持精度不变和能够有效地处理大的数据集；可以处理没有删减的成千上万的变量；能够在分类的过程中可以生成一个泛化误差的内部无偏估计；能够检测到特征之间的相互影响以及重要性程度；不过出现过度拟合；实现简单容易并行化。

**1. 决策树的基本认识**

**决策树**是一种依托决策而建立起来的一种树。在[**机器学习**](http://lib.csdn.net/base/machinelearning)中，决策树是一种预测模型，代表的是一种对

   象属性与对象值之间的一种映射关系，每一个节点代表某个对象，树中的每一个分叉路径代表某个可能

   的属性值，而每一个叶子节点则对应从根节点到该叶子节点所经历的路径所表示的对象的值。决策树仅

   有单一输出，如果有多个输出，可以分别建立独立的决策树以处理不同的输出。接下来讲解ID3算法。

**2. ID3算法介绍**

**ID3算法**是决策树的一种，它是基于**奥卡姆剃刀原理**的，即用尽量用较少的东西做更多的事。**ID3算法**，

   即**Iterative Dichotomiser 3**，**迭代二叉树3代**，是Ross Quinlan发明的一种决策树算法，这个

   算法的基础就是上面提到的奥卡姆剃刀原理，越是小型的决策树越优于大的决策树，尽管如此，也不总

   是生成最小的树型结构，而是一个启发式算法。

   在信息论中，期望信息越小，那么信息增益就越大，从而纯度就越高。ID3算法的核心思想就是以信息

   增益来度量属性的选择，选择分裂后信息增益最大的属性进行分裂。该算法采用自顶向下的贪婪搜索遍

   历可能的决策空间。

**3. 信息熵与信息增益**

   在信息增益中，重要性的衡量标准就是看特征能够为分类系统带来多少信息，带来的信息越多，该特征越

   重要。在认识信息增益之前，先来看看**信息熵**的定义

**熵**这个概念最早起源于物理学，在物理学中是用来度量一个热力学系统的无序程度，而在信息学里面，熵

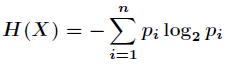
   是对不确定性的度量。在**1948**年，**香农**引入了**信息熵**，将其定义为离散随机事件出现的概率，一个系统越

   是有序，信息熵就越低，反之一个系统越是混乱，它的信息熵就越高。所以信息熵可以被认为是系统有序

   化程度的一个度量。

   假如一个随机变量http://images.cnitblog.com/blog/571227/201412/112108114467870.png的取值为http://images.cnitblog.com/blog/571227/201412/112108491346383.png，每一种取到的概率分别是http://images.cnitblog.com/blog/571227/201412/112110282909943.png，那么

http://images.cnitblog.com/blog/571227/201412/112110572751166.png的熵定义为

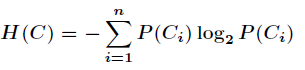


   意思是一个变量的变化情况可能越多，那么它携带的信息量就越大。

   对于**分类系统**来说，类别http://images.cnitblog.com/blog/571227/201412/112119378211796.png是变量，它的取值是http://images.cnitblog.com/blog/571227/201412/112120381502926.png，而每一个类别出现的概率分别是

http://images.cnitblog.com/blog/571227/201412/112121283685834.png

   而这里的http://images.cnitblog.com/blog/571227/201412/112122176655828.png就是类别的总数，此时分类系统的熵就可以表示为

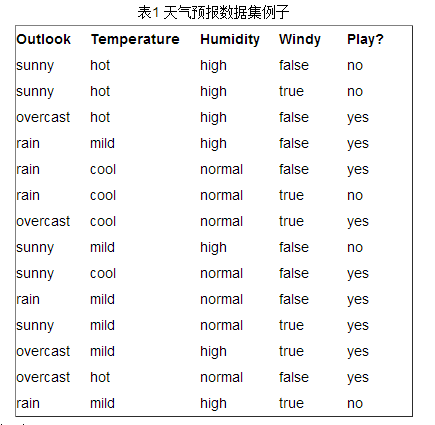


   以上就是信息熵的定义，接下来介绍**信息增益**。

   信息增益是针对一个一个特征而言的，就是看一个特征http://images.cnitblog.com/blog/571227/201412/121352330406260.png，系统有它和没有它时的信息量各是多少，两者

   的差值就是这个特征给系统带来的信息量，即**信息增益**。

   接下来以**天气预报**的例子来说明。下面是描述天气数据表，学习目标是**play**或者**not play**。

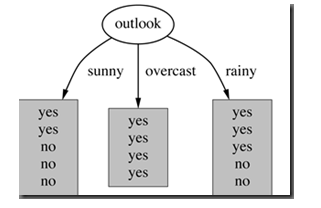


   可以看出，一共**14**个样例，包括**9**个正例和**5**个负例。那么当前信息的熵计算如下

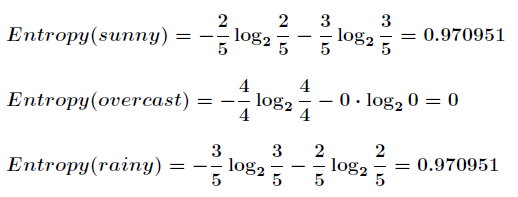
http://images.cnitblog.com/blog/571227/201412/121404563217970.png

   在决策树分类问题中，信息增益就是决策树在进行属性选择划分前和划分后信息的差值。假设利用

   属性**Outlook**来分类，那么如下图



      划分后，数据被分为三部分了，那么各个分支的信息熵计算如下



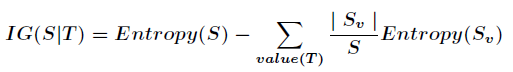
       那么划分后的信息熵为

http://images.cnitblog.com/blog/571227/201412/121424414622037.png

http://images.cnitblog.com/blog/571227/201412/121425239464583.png代表在特征属性http://images.cnitblog.com/blog/571227/201412/121426460096486.png的条件下样本的**条件熵**。那么最终得到特征属性http://images.cnitblog.com/blog/571227/201412/121431418067920.png带来的信息增益为

http://images.cnitblog.com/blog/571227/201412/121433352752920.png

**信息增益的计算公式**如下



   其中http://images.cnitblog.com/blog/571227/201412/121630415566687.png为全部样本集合，http://images.cnitblog.com/blog/571227/201412/121632460875366.png是属性http://images.cnitblog.com/blog/571227/201412/121633238687609.png所有取值的集合，http://images.cnitblog.com/blog/571227/201412/121634348377590.png是http://images.cnitblog.com/blog/571227/201412/121634497753388.png的其中一个属性值，http://images.cnitblog.com/blog/571227/201412/121635324782234.png是http://images.cnitblog.com/blog/571227/201412/121635438217699.png中属性http://images.cnitblog.com/blog/571227/201412/121636047127813.png的

   值为http://images.cnitblog.com/blog/571227/201412/121636324151078.png的样例集合，http://images.cnitblog.com/blog/571227/201412/121637243376773.png为http://images.cnitblog.com/blog/571227/201412/121637359468253.png中所含样例数。

   在决策树的每一个非叶子结点划分之前，先计算每一个属性所带来的信息增益，选择最大信息增益的属性来划

   分，因为信息增益越大，区分样本的能力就越强，越具有代表性，很显然这是一种自顶向下的贪心策略。以上

   就是**ID3算法**的核心思想。

**C4.5**[**算法**](http://lib.csdn.net/base/datastructure)是[**机器学习**](http://lib.csdn.net/base/machinelearning)中的一个重要的决策树算法，它是对ID3算法的改进，相对于ID3算法主要有以下几个改进

  （1）用信息增益率来选择属性

  （2）在决策树的构造过程中对树进行剪枝

  （3）对非离散数据也能处理

  （4）能够对不完整数据进行处理

接下来分别详细讲述这几点的改进方案

**（1）用信息增益率来选择属性**

    在ID3算法中，我们知道是用信息增益来选择属性的，而信息增益的缺点是比较偏向选择取值较多的属性，

    在C4.5算法中，除了一项分裂信息来惩罚取值更多的属性，所以得到如下公式

http://images.cnitblog.com/blog/571227/201501/241741574538672.png

    其中http://images.cnitblog.com/blog/571227/201501/241742118916384.png表示信息增益，而http://images.cnitblog.com/blog/571227/201501/241742256887266.png表示分裂信息，它的计算公式如下

http://images.cnitblog.com/blog/571227/201501/241743089222442.png

http://images.cnitblog.com/blog/571227/201501/241745273288557.png表示当前属性的所有取值。

**（2）在决策树的构造过程中对树进行剪枝**

    在C4.5算法中，采用了**悲观剪枝**的方法，它使用训练集生成决策树，又用训练集来进行剪枝。

    悲观剪枝法的基本原理参考：[**http://www.cnblogs.com/zhangchaoyang/articles/2842490.html**](http://www.cnblogs.com/zhangchaoyang/articles/2842490.html)

**（3）对非离散数据也能处理**

    其实C4.5算法对连续性数据的处理也是当作离散数据处理的，具体可以参考上面的链接。

**1. CART算法的认识**

**Classification And Regression Tree**，即**分类回归树**算法，简称**CART算法**，它是决策树的一种实现，通

   常决策树主要有三种实现，分别是ID3算法，CART算法和C4.5算法。

   CART算法是一种二分递归分割技术，把当前样本划分为两个子样本，使得生成的每个非叶子结点都有两个分支，

   因此CART算法生成的决策树是结构简洁的二叉树。由于CART算法构成的是一个二叉树，它在每一步的决策时只能

   是“是”或者“否”，即使一个feature有多个取值，也是把数据分为两部分。在CART算法中主要分为两个步骤

   （1）将样本递归划分进行建树过程

   （2）用验证数据进行剪枝

**2. CART算法的原理**

   上面说到了CART算法分为两个过程，其中第一个过程进行递归建立二叉树，那么它是如何进行划分的 ？

   设http://images.cnitblog.com/blog/571227/201501/091521164535431.png代表单个样本的http://images.cnitblog.com/blog/571227/201501/091522035317783.png个属性，http://images.cnitblog.com/blog/571227/201501/091522421257425.png表示所属类别。CART算法通过递归的方式将http://images.cnitblog.com/blog/571227/201501/091523317818089.png维的空间划分为不重

   叠的矩形。划分步骤大致如下

   （1）选一个自变量http://images.cnitblog.com/blog/571227/201501/091528272651984.png，再选取http://images.cnitblog.com/blog/571227/201501/091531419535737.png的一个值http://images.cnitblog.com/blog/571227/201501/091532214533937.png，http://images.cnitblog.com/blog/571227/201501/091532442966491.png把http://images.cnitblog.com/blog/571227/201501/091533267036202.png维空间划分为两部分，一部分的所有点都满足http://images.cnitblog.com/blog/571227/201501/091534271563502.png，

       另一部分的所有点都满足http://images.cnitblog.com/blog/571227/201501/091535186096649.png，对非连续变量来说属性值的取值只有两个，即等于该值或不等于该值。

   （2）递归处理，将上面得到的两部分按步骤（1）重新选取一个属性继续划分，直到把整个http://images.cnitblog.com/blog/571227/201501/091536471093412.png维空间都划分完。

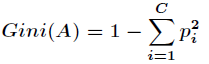
   在划分时候有一个问题，它是按照什么标准来划分的 ？ 对于一个变量属性来说，它的划分点是一对连续变量属

   性值的中点。假设http://images.cnitblog.com/blog/571227/201501/091549258753804.png个样本的集合一个属性有http://images.cnitblog.com/blog/571227/201501/091550106258598.png个连续的值，那么则会有http://images.cnitblog.com/blog/571227/201501/091551330314125.png个分裂点，每个分裂点为相邻

   两个连续值的均值。每个属性的划分按照能减少的杂质的量来进行排序，而杂质的减少量定义为划分前的杂质减

   去划分后的每个节点的杂质量划分所占比率之和。而杂质度量方法常用**Gini指标**，假设一个样本共有http://images.cnitblog.com/blog/571227/201501/091603514532531.png类，那么

   一个节点http://images.cnitblog.com/blog/571227/201501/091604407344010.png的Gini不纯度可定义为



   其中http://images.cnitblog.com/blog/571227/201501/091606025624178.png表示属于http://images.cnitblog.com/blog/571227/201501/091606342504996.png类的概率，当Gini(A)=0时，所有样本属于同类，所有类在节点中以等概率出现时，Gini(A)

   最大化，此时http://images.cnitblog.com/blog/571227/201501/091608573281118.png。

   有了上述理论基础，实际的递归划分过程是这样的：如果当前节点的所有样本都不属于同一类或者只剩下一个样

   本，那么此节点为非叶子节点，所以会尝试样本的每个属性以及每个属性对应的分裂点，尝试找到杂质变量最大

   的一个划分，该属性划分的子树即为最优分支。

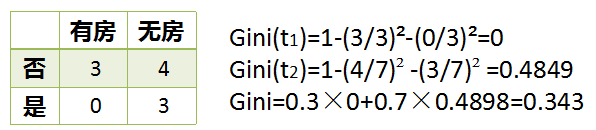
   下面举个简单的例子，如下图



   在上述图中，属性有3个，分别是有房情况，婚姻状况和年收入，其中有房情况和婚姻状况是离散的取值，而年

   收入是连续的取值。拖欠贷款者属于分类的结果。

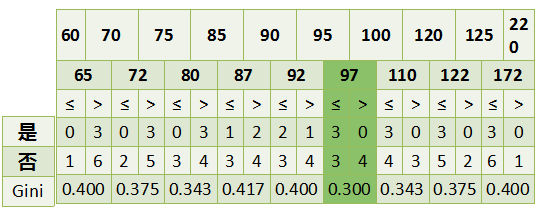
   假设现在来看有房情况这个属性，那么按照它划分后的Gini指数计算如下



   而对于婚姻状况属性来说，它的取值有3种，按照每种属性值分裂后Gini指标计算如下



   最后还有一个取值连续的属性，年收入，它的取值是连续的，那么连续的取值采用分裂点进行分裂。如下



   根据这样的分裂规则CART算法就能完成建树过程。

   建树完成后就进行第二步了，即根据验证数据进行剪枝。在CART树的建树过程中，可能存在Overfitting，许多

   分支中反映的是数据中的异常，这样的决策树对分类的准确性不高，那么需要检测并减去这些不可靠的分支。决策

   树常用的剪枝有事前剪枝和事后剪枝，CART算法采用事后剪枝，具体方法为**代价复杂性剪枝法**。可参考如下链接

前面介绍过决策树的三种实现：**ID3**[**算法**](http://lib.csdn.net/base/datastructure)，**C4.5算法**和**CART算法**。虽然这些决策树有很多优良的性质，比如训练时间

复杂度较低，模型容易展示等等，但是同时单决策树有一些不好的地方，比如容易over-fitting，虽然剪枝可以减

少这种现象的发生，但是还是不够的。为了减少决策树的不足，近年来又提出了许多**模型组和+决策树**的算法，这些算

法都是生成N棵决策树，虽然这N棵树都很简单，但是它们综合起来就很强大了。今天就来介绍**随机森林**。

**Contents**

**1. 随机森林的基本原理**

**2. 随机森林的具体步骤**

**3. 随机森林的优点**

**4. 随机森林开源框架**

**1. 随机森林的基本原理**

   在[**机器学习**](http://lib.csdn.net/base/machinelearning)中，随机森林是一个包含多个决策树的分类器，并且其输出的类型是由每个树输出类别的**众数**而定。顾

   名思义，随机森林就是用随机的方式构建一个森林，这个森林由很多的决策树构成，随机森林的每棵决策树之间是

   没有关联的。在建好随机森林后，当有一个新的样本输入后，就让森林中的每棵决策树都进行判断，最终的预测结

   果就是这N棵决策树的众数对应的种类。

**2. 随机森林的具体步骤**

   在随机森林中，最重要的是如何构造一个随机森林。假设数据样本数为N，那么每棵决策树采样的样本数也就是N，

   每个样本的属性个数为M，在每个决策树构造过程中，每个节点随机选择m个属性计算最佳分裂方式进行分裂。具

   体步骤如下

   （1）有放回地随机选择N个样本，用这N个样本来训练一棵决策树。

   （2）每个样本有M个属性，在决策树中需要分裂节点时，从这M个属性中随机选取m个属性，一般来说m << M，

       然后从这m个属性中采用某种策略选择最佳属性作为当前节点的分裂属性。

   （3）每棵决策树的每个节点的分裂都按照步骤（2）进行，直到不能分裂为止。

   （4）对于每棵决策树都这样建立，就得到了随机森林。

   随机森林的随机性体现在每棵树的训练样本是随机的，树中每个节点的分裂属性也是随机选择的。有了这2个随机

   因素，即使每棵决策树没有进行剪枝，随机森林也不会产生过拟合的现象。

   随机森林中有两个人为控制参数：**森林中树的数量**（一般选取值较大）和**m值的大小**（一般选取为M的平方根）。

**3. 随机森林的优点**

   随机森林有很多优点，具体如下

   （1）分类结果更加准确

   （2）可以处理高维度的属性，并且不用做特征选择

   （3）即使有很大部分数据遗失，仍可以维持高准确度

   （4）学习过程快速

   （5）在训练完成后，能够给出哪些属性比较重要

   （6）容易实现并行化计算

   （7）在训练过程中，能够检测到属性之间的相互影响

**HMM（隐马尔科夫模型）在股票上的简单应用**

[](http://www.jianshu.com/u/4fbdf93443fd)

作者 [Ricequant](http://www.jianshu.com/u/4fbdf93443fd) 关注

2016.04.06 21:01\* 字数 2278 阅读 1799评论 0喜欢 12

**Ricequant团队出品，如需转发请注明且请私信联系，否则必究。**

**原文：**[**https://www.ricequant.com/community/topic/788/**](https://www.ricequant.com/community/topic/788/)

今天我们来介绍一下**HMM**（**隐马尔科夫模型**）在股票上的简单应用。

**隐马尔科夫**模型，乍一听起来好高端，完全不知道是什么鬼，那么就让我们退一步，先看看**马尔科夫链**。

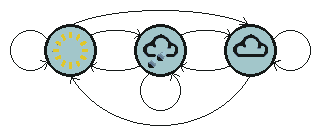
马尔可夫链，因安德烈·马尔可夫（A.A.Markov，1856－1922）得名（就是下面这家伙），是指数学中具有**马尔可夫性质**的离散事件随机过程。在给定当前知识或信息的情况下，过去（即当前以前的历史状态）对于预测将来（即当前以后的未来状态）是无关的。



该过程中，每个状态的转移只依赖于之前的n个状态，这个过程被称为1个n阶的模型，其中n是影响转移状态的数目。最简单的**马尔科夫过程**就是一阶过程，每一个状态的转移只依赖于其之前的那一个状态。

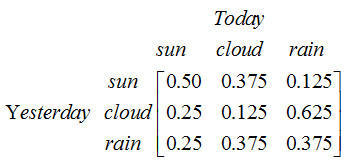
用数学表达式表示就是下面的样子：

举一个日常生活中的例子，我们希望根据当前天气的情况来预测未来天气情况。一种办法就是假设这个模型的每个状态都只依赖于前一个的状态，即马尔科夫假设，这个假设可以极大简化这个问题。当然，这个例子也是有些不合实际的。但是，这样一个简化的系统可以有利于我们的分析，所以我们通常接受这样的假设，因为我们知道这样的系统能让我们获得一些有用的信息，尽管不是十分准确的。上面的图显示了天气进行转移的模型。

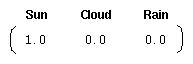


http://upload-images.jianshu.io/upload_images/1865974-816edfbd246cf52e.jpg?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240

注意一个含有N个状态的一阶过程有N²个状态转移。每一个转移的概率叫做**状态转移概率**，就是从一个状态转移到另一个状态的概率。这所有的N²个概率可以用一个状态转移矩阵来表示，上面天气例子的状态转移矩阵如下：这个矩阵表示，如果昨天是阴天，那么今天有25%的可能是晴天，12.5%的概率是阴天，62.5%的概率会下雨，很明显，矩阵中每一行的和都是1。



为了初始化这样一个系统，我们需要一个初始的概率向量：



这个向量表示第一天是晴天。到这里，我们就为上面的一阶马尔科夫过程定义了以下三个部分：

状态：晴天、阴天和下雨。

初始向量：定义系统在时间为0的时候的状态的概率。

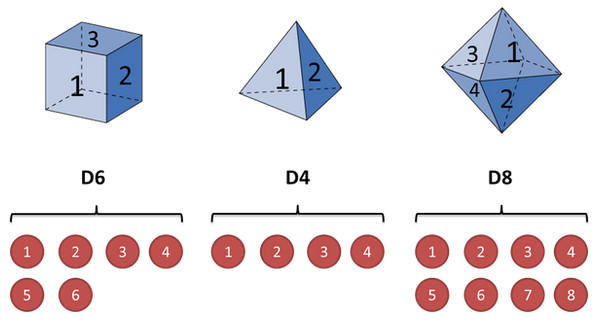
状态转移矩阵：每种天气转换的概率。所有的能被这样描述的系统都是一个**马尔科夫过程**。

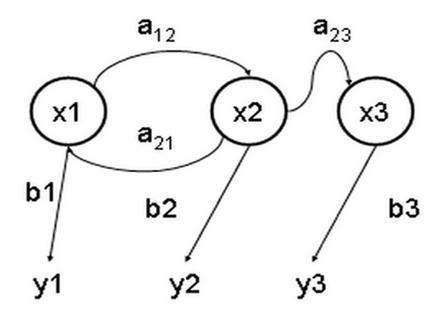
然而，当马尔科夫过程不够强大的时候，我们又该怎么办呢？在某些情况下，马尔科夫过程不足以描述我们希望发现的模式。

比如我们的股市，如果只是观测市场，我们只能知道当天的价格、成交量等信息，但是并不知道当前股市处于什么样的状态（牛市、熊市、震荡、反弹等等），在这种情况下我们有两个状态集合，一个可以观察到的状态集合（股市价格成交量状态等）和一个隐藏的状态集合（股市状况）。我们希望能找到一个算法可以根据股市价格成交量状况和马尔科夫假设来预测股市的状况。

在上面的这些情况下，可以观察到的**状态序列和隐藏的状态序列是概率相关**的。于是我们可以将这种类型的过程建模为有一个隐藏的马尔科夫过程和一个与这个隐藏马尔科夫过程概率相关的并且可以观察到的状态集合，就是**隐马尔可夫模型**。

**隐马尔可夫模型**(Hidden Markov Model) 是一种统计模型，用来描述一个含有隐含未知参数的马尔可夫过程。其难点是从可观察的参数中确定该过程的隐含参数，然后利用这些参数来作进一步的分析。下图是一个三个状态的隐马尔可夫模型状态转移图，其中x表示隐含状态，y表示可观察的输出，a表示状态转换概率，b表示输出概率。用一个掷筛子的例子阐述一下：假设我手里有三个不同的骰子。第一个骰子是我们平常见的骰子（称这个骰子为D6），6个面，每个面（1，2，3，4，5，6）出现的概率是1/6。第二个骰子是个四面体（称这个骰子为D4），每个面（1，2，3，4）出现的概率是1/4。第三个骰子有八个面（称这个骰子为D8），每个面（1，2，3，4，5，6，7，8）出现的概率是1/8。



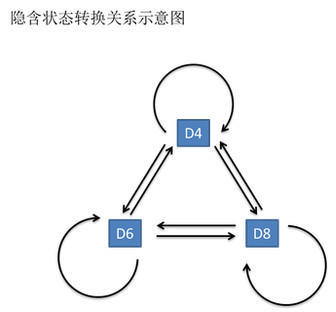


假设我们开始掷骰子，我们先从三个骰子里挑一个，挑到每一个骰子的概率都是1/3。然后我们掷骰子，得到一个数字，1，2，3，4，5，6，7，8中的一个。不停的重复上述过程，我们会得到一串数字，每个数字都是1，2，3，4，5，6，7，8中的一个。例如我们可能得到这么一串数字（掷骰子10次）：1 6 3 5 2 7 3 5 2 4

这串数字叫做可见状态链。但是在隐马尔可夫模型中，我们不仅仅有这么一串可见状态链，还有一串隐含状态链。在这个例子里，这串隐含状态链就是你用的骰子的序列。比如，隐含状态链有可能是：D4 D6 D8 D6 D4 D8 D6 D6 D6 D4。

一般来说，HMM中说到的马尔可夫链其实是指隐含状态链，因为隐含状态（骰子）之间存在转换概率。在我们这个例子里，D6的下一个状态是D4，D6，D8的概率都是1/3。D4，D8的下一个状态是D4，D6，D8的转换概率也都一样是1/3。这样设定是为了最开始容易说清楚，但是我们其实是可以随意设定转换概率的。比如，我们可以这样定义，D6后面不能接D4，D6后面是D6的概率是0.9，是D8的概率是0.1。这样就是一个新的HMM。

同样的，尽管可见状态之间没有转换概率，但是隐含状态和可见状态之间有一个概率叫做输出概率。就我们的例子来说，六面骰子（D6）产生1的输出概率是1/6。产生2，3，4，5，6的概率也都是1/6。我们同样可以对输出概率进行其他定义。比如，我有一个被赌场动过手脚的六面骰子，掷出来是1的概率更大，是1/2，掷出来是2，3，4，5，6的概率是1/10。



其实对于HMM来说，如果提前知道所有隐含状态之间的转换概率和所有隐含状态到所有可见状态之间的输出概率，做模拟是相当容易的。但是应用HMM模型时候呢，往往是缺失了一部分信息的，有时候你知道骰子有几种，每种骰子是什么，但是不知道掷出来的骰子序列；有时候你只是看到了很多次掷骰子的结果，剩下的什么都不知道。如果应用算法去估计这些缺失的信息，就成了一个很重要的问题。

和HMM模型相关的算法主要分为三类，分别解决三种问题：

知道骰子有几种（隐含状态数量），每种骰子是什么（转换概率），根据掷骰子掷出的结果（可见状态链），我想知道每次掷出来的都是哪种骰子（隐含状态链）。

还是知道骰子有几种（隐含状态数量），每种骰子是什么（转换概率），根据掷骰子掷出的结果（可见状态链），我想知道掷出这个结果的概率。

知道骰子有几种（隐含状态数量），不知道每种骰子是什么（转换概率），观测到很多次掷骰子的结果（可见状态链），我想反推出每种骰子是什么（转换概率）。

如果要解决上面股市中的问题，我们就需要解决问题1和问题3，下面我们就看看如何实现。