数据挖掘入门

0.9 版

编著
http://datamining.126.com

相关读物

M. Berry and G. Linoff, *Data Mining Techniques,* John Wiley, 1997
William S. Cleveland, *The Elements of Graphing Data, revised,* Hobart Press, 1994
Howard Wainer, *Visual Revelations,* Copernicus, 1997
R. Kennedy, Lee, Reed, and Van Roy, *Solving Pattern Recognition Problems,* Prentice-Hall, 1998
U. Fayyad, Piatetsky-Shapiro, Smyth, and Uthurusamy, *Advances in Knowledge Discovery and Data Mining,* MIT Press, 1996
Dorian Pyle, *Data Preparation for Data Mining,* Morgan Kaufmann, 1999
C. Westphal and T. Blaxton, *Data Mining Solutions,* John Wiley, 1998
Vasant Dhar and Roger Stein, *Seven Methods for Transforming Corporate Data into Business Intelligence,* Prentice Hall 1997
Brieman, Freidman, Olshen, and Stone, *Classification and Regression Trees,* Wadsworth, 1984
J. R. Quinlan, *C4.5: Programs for Machine Learning,* Morgan Kaufmann, 1992

目录

介绍

什么是数据挖掘

当今数据库的容量已经达到上万亿的水平（T）--- 1,000,000,000,000 个字节。在这些大量数据的背后隐藏了很多具有决策意义的信息，那么怎么得到这些“知识”呢？也就是怎样通过一颗颗的树木了解到整个森林的情况？

计算机科学对这个问题给出的最新回答就是：数据挖掘，在“数据矿山”中找到蕴藏的“知识金块”，帮助企业减少不必要投资的同时提高资金回报。数据挖掘给企业带来的潜在的投资回报几乎是无止境的。世界范围内具有创新性的公司都开始采用数据挖掘技术来判断哪些是他们的最有价值客户、重新制定他们的产品推广策略（把产品推广给最需要他们的人），以用最小的花费得到最好的销售。

数据挖掘是一个利用各种分析工具在海量数据中发现模型和数据间关系的过程，这些模型和关系可以用来做出预测。

数据挖掘的第一步是**描述**数据 --- 计算统计变量（比如平均值、均方差等），再用图表或图片直观的表示出来，进而可以看出一些变量之间的相关性（比如有一些值经常同时出现）。选择正确的数据源对整个数据挖掘项目的成败至关重要，在后面*数据挖掘的步骤*中我们会着重强调这一点。

单单是数据描述并不能为人们制订行动计划提供足够的依据，你必须用你的这些历史数据**建立一个预言模型**，然后再用另外一些数据对这个模型进行**测试**。一个好的模型没必要与数据库中的数据 100%的相符（城市交通图也不是完全的实际交通线路的等比缩小），但他在你做决策时是一个很好的指南和依据。

最后一步是**验证**你的模型。比如你用所有对你的产品推广计划做出回应的人的数据库做了一个模型，来预测什么样的人会对你的产品感兴趣。你能在得到这个模型后就直接利用这个模型做出决策或采取行动吗？还是更稳妥一点先对一小部分客户做一个实际的测试，然后再决定？

数据挖掘：不能干什么

数据挖掘是一个工具，而不是有魔力的权杖。它不会坐在你的数据库上一直监视着数据库，然后当他发现有意义的模型时给你发一封电子邮件。他仍然需要了解你的业务，理解你的数据，弄清分析方法。数据挖掘只是帮助商业人士更深入、更容易的分析数据 --- 他无法告诉你某个模型对你的企业的实际价值。而且数据挖掘中得到的模型必须要在现实生活中进行验证。

注意数据挖掘中得到的预言模型并不会告诉你一个人为什么会做一件事、采取某个行动，他只会告诉你他**会**这样做，**为什么**要人去考虑。比如，数据挖掘可能会告诉你，如果这个人是男的、年收入在 5 万到 6 万之间，那么他可能会买你的商品/服务。你可能会利用这条规则，集中向这类人推销你的商品而从中获益，但是数据挖掘工具不会告诉你他们为什么会买你的东西，也不能保证所有符合这条规则的人都会买。

为了保证数据挖掘结果的价值，你自己必须了解你的数据，这一点至关重要。输入数据库中的异常数据、不相关的字段或互相冲突的字段（比如年龄和生日不一致）、数据的编码方式等都会对数据挖掘输出结果的质量产生影响。虽然一些算法自身会对上面提到的这些问题做一些考虑，但让算法自己做所有这些决定是不明智的。

数据挖掘不会在缺乏指导的情况下自动的发现模型。你不能这样对数据挖掘工具说，“帮我提高直接邮件推销的响应率”，你应该让数据挖掘工具找(1)对你的推销回应的人，或(2)即回应又做了大量订单的人的特征。在数据挖掘中寻找这两种模型是很不相同的。

虽然数据挖掘工具使你不必再掌握艰深的统计分析技术，但你仍然需要知道你所选用的数据挖掘工具是如何工作的，他所采用的算法的原理是什么。你所选用的技术和优化方法会对你的模型的准确度和生成速度产生很大影响。

数据挖掘永远不会替代有经验的商业分析师或管理人员所起的作用，他只是提供一个强大的工具。每个成熟的、了解市场的公司都已经具有一些重要的、能产生高回报的模型，这些模型可能是管理人员花了很长时间，作了很多调查，甚至是经过很多失误之后得来的。数据挖掘工具要做的就是使这些模型得到的更容易，更方便，而且有根据。

数据挖掘和数据仓库

大部分情况下，数据挖掘都要先把数据从数据仓库中拿到数据挖掘库或数据集市中（见图 1）。从数据仓库中直接得到进行数据挖掘的数据有许多好处。就如我们后面会讲到的，数据仓库的数据清理和数据挖掘的数据清理差不多，如果数据在导入数据仓库时已经清理过，那很可能在做数据挖掘时就没必要在清理一次了，而且所有的数据不一致的问题都已经被你解决了。

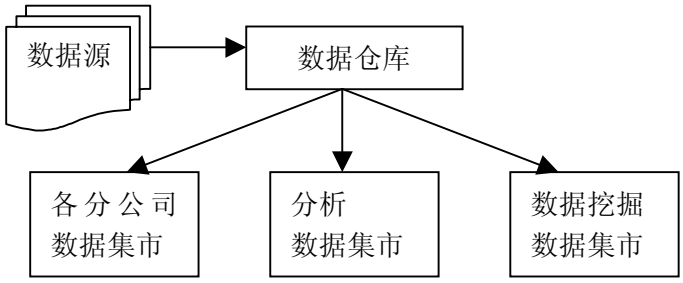

图 1：数据挖掘库从数据仓库中得出

数据挖掘库可能是你的数据仓库的一个逻辑上的子集，而不一定非得是物理上单独的数据库。但如果你的数据仓库的计算资源已经很紧张，那你最好还是建立一个单独的数据挖掘库。

当然为了数据挖掘你也不必非得建立一个数据仓库，数据仓库不是必需的。建立一个巨大的数据仓库，把各个不同源的数据统一在一起，解决所有的数据冲突问题，然后把所有的数据导到一个数据仓库内，是一项巨大的工程，可能要用几年的时间花上百万的钱才能完成。只是为了数据挖掘，你可以把一个或几个事务数据库导到一个只读的数据库中，就把它当作数据集市，然后在他上面进行数据挖掘。

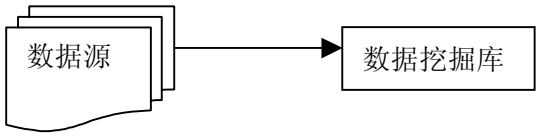

图 2：数据挖掘库从事务数据库中得出

数据挖掘和在线分析处理（OLAP）

一个经常问的问题是，数据挖掘和 OLAP 到底有何不同。下面将会解释，他们是完全不同的工具，基于的技术也大相径庭。

OLAP 是决策支持领域的一部分。传统的查询和报表工具是告诉你数据库中都有什么（what happened），OLAP 则更进一步告诉你下一步会怎么样（What next）、和如果我采取这样的措施又会怎么样（What if）。用户首先建立一个假设，然后用 OLAP 检索数据库来验证这个假设是否正确。比如，一个分析师想找到什么原因导致了贷款拖欠，他可能先做一个初始的假定，认为低收入的人信用度也低，然后用 OLAP 来验证他这个假设。如果这个假设没有被证实，他可能去察看那些高负债的账户，如果还不行，他也许要把收入和负债一起考虑，一直进行下去，直到找到他想要的结果或放弃。

也就是说，OLAP 分析师是建立一系列的假设，然后通过 OLAP 来证实或推翻这些假设来最终得到自己的结论。OLAP 分析过程在本质上是一个演绎推理的过程。但是如果分析的变量达到几十或上百个，那么再用 OLAP 手动分析验证这些假设将是一件非常困难和痛苦的事情。

数据挖掘与 OLAP 不同的地方是，数据挖掘不是用于验证某个假定的模式（模型）的正确性，而是在数据库中自己寻找模型。他在本质上是一个归纳的过程。比如，一个用数据挖掘工具的分析师想找到引起贷款拖欠的风险因素。数据挖掘工具可能帮他找到高负债和低收入是引起这个问题的因素，甚至还可能发现一些分析师从来没有想过或试过的其他因素，比如年龄。

数据挖掘和 OLAP 具有一定的互补性。在利用数据挖掘出来的结论采取行动之前，你也许要验证一下如果采取这样的行动会给公司带来什么样的影响，那么 OLAP 工具能回答你的这些问题。

而且在知识发现的早期阶段，OLAP 工具还有其他一些用途。可以帮你探索数据，找到哪些是对一个问题比较重要的变量，发现异常数据和互相影响的变量。这都能帮你更好的理解你的数据，加快知识发现的过程。

数据挖掘，机器学习和统计

数据挖掘利用了人工智能（AI）和统计分析的进步所带来的好处。这两门学科都致力于模式发现和预测。

数据挖掘不是为了替代传统的统计分析技术。相反，他是统计分析方法学的延伸和扩展。大多数的统计分析技术都基于完善的数学理论和高超的技巧，预测的准确度还是令人满意的，但对使用者的要求很高。而随着计算机计算能力的不断增强，我们有可能利用计算机强大的计算能力只通过相对简单和固定的方法完成同样的功能。

一些新兴的技术同样在知识发现领域取得了很好的效果，如神经元网络和决策树，在足够多的数据和计算能力下，他们几乎不用人的关照自动就能完成许多有价值的功能。

数据挖掘就是利用了统计和人工智能技术的应用程序，他把这些高深复杂的技术封装起来，使人们不用自己掌握这些技术也能完成同样的功能，并且更专注于自己所要解决的问题。

软硬件发展对数据挖掘的影响

使数据挖掘这件事情成为可能的关键一点是计算机性能价格比的巨大进步。在过去的几年里磁盘存储器的价格几乎降低了 99%，这在很大程度上改变了企业界对数据收集和存储的态度。如果每兆的价格是￥10，那存放 1TB 的价格是￥10,000,000，但当每兆的价格降为 1 毛钱时，存储同样的数据只有￥100,000！

计算机计算能力价格的降低同样非常显著。每一代芯片的诞生都会把 CPU 的计算能力提高一大步。内存 RAM 也同样降价迅速，几年之内每兆内存的价格由几百块钱降到现在只要几块钱。通常 PC 都有 64M 内存，工作站达到了 256M，拥有上 G 内存的服务器已经不是什么新鲜事了。

在单个CPU计算能力大幅提升的同时，基于多个CPU的并行系统也取得了很大的进步。目前几乎所有的服务器都支持多个 CPU，这些 SMP 服务器簇甚至能让成百上千个 CPU 同时工作。

基于并行系统的数据库管理系统也给数据挖掘技术的应用带来了便利。如果你有一个庞大而复杂的数据挖掘问题要求通过访问数据库取得数据，那么效率最高的办法就是利用一个本地的并行数据库。

所有这些都为数据挖掘的实施扫清了道路，随着时间的延续，我们相信这条道路会越来越平坦。

数据挖掘应用

由于数据挖掘带来的显著的经济效益，使数据挖掘越来越普及。他不仅能用于控制成本，也能给企业带来效益。

很多企业都在利用数据挖掘技术帮助管理客户生命周期的各个阶段，包括争取新的客户、在已有客户的身上赚更多的钱、和保持住好的客户。如果能够确定好的客户的特点，那么就能提供为客户提供针对性的服务。比如，已经发现了购买某一商品的客户的特征，那么就可以向那些具有这些特征但还没有购买此商品的客户推销这个商品；找到流失的客户的特征就可以，在那些具有相似特征的客户还未流失之前进行针对性的弥补，因为保留一个客户要比争取一个客户便宜的多。

数据挖掘可以应用在各个不同的领域。电讯公司和信用卡公司是用数据挖掘检测欺诈行为的先行者。保险公司和证券公司也开始采用数据挖掘来减少欺诈。医疗应用是另一个前景广阔的产业：数据挖掘可以用来预测外科手术、医疗试验和药物治疗的效果。零销商更多的使用数据挖掘来决定每种商品在不同地点的库存，通过数据挖掘更灵活的使用促销和优惠卷手段。制药公司通过挖掘巨大的化学物质和基因对疾病的影响的数据库来判断哪些物质可能对治疗某种疾病产生效果。

成功的数据挖掘

有保证数据挖掘成功的两个关键要素。一是准确的定义你所要解决的问题，定位准确的问题通常会带来最好的回报。二是使用正确的数据，选定了你所能得到的数据，也许还要从外部购买数据，你需要对这些数据做有效的数据整合和转换。

描述型数据挖掘

统计和可视化

要想建立一个好的预言模型，你必须了解自己的数据。最基本的方法是计算各种统计变量（平均值、方差等）和察看数据的分布情况。你也可以用数据透视表察看多维数据。

数据的种类可分为*连续的*，有一个用数字表示的值（比如销售量）或*离散的*，分成一个个的类别（如红、绿、蓝）。离散数据可以进一步分为*可排序的*，数据间可以比较大小（如，高、中、低）和*标称的*，不可排序（如邮政编码）。

图形和可视化工具在数据准备阶段尤其重要，它能让你快速直观的分析数据，而不是给你枯燥乏味的文本和数字。它不仅让你看到整个森林，还允许你拉近每一棵树来察看细节。在图形模式下人们很容易找到数据中可能存在的模式、关系、异常等，直接看数字则很难。

可视化工具的问题是模型可能有很多维或变量，但是我们只能在 2 维的屏幕或纸上展示它。比如，我们可能要看的是信用风险与年龄、性别、婚姻状况、参加工作时间的关系。因此，可视化工具必须用比较巧妙的方法在两维空间内展示 n 维空间的数据。虽然目前有了一些这样的工具，但它们都要用户“训练”过他们的眼睛后才能理解图中画的到底是什么东西。对于眼睛有色盲或空间感不强的人，在使用这些工具时可能会遇到困难。

聚集（分群）

聚集是把整个数据库分成不同的群组。它的目的是要群与群之间差别很明显，而同一个群之间的数据尽量相似。与分类不同（见后面的预测型数据挖掘），在开始聚集之前你不知道要把数据分成几组，也不知道怎么分（依照哪几个变量）。因此在聚集之后要有一个对业务很熟悉的人来解释这样分群的意义。很多情况下一次聚集你得到的分群对你的业务来说可能并不好，这时你需要删除或增加变量以影响分群的方式，经过几次反复之后才能最终得到一个理想的结果。神经元网络和 K-均值是比较常用的聚集算法。

不要把聚集与分类混淆起来。在分类之前，你已经知道要把数据分成哪几类，每个类的性质是什么，聚集则恰恰相反。

关联分析

关联分析是寻找数据库中值的相关性。两种常用的技术是*关联规则*和*序列模式*。关联规则是寻找在同一个事件中出现的不同项的相关性，比如在一次购买活动中所买不同商品的相关性。序列模式与此类似，他寻找的是事件之间时间上的相关性，如对股票涨跌的分析。

关联规则可记为 A==>B，A 称为前提和左部（LHS），B 称为后续或右部（RHS）。如关联规则“买锤子的人也会买钉子”，左部是“买锤子”，右部是“买钉子”。

要计算包含某个特定项或几个项的事务在数据库中出现的概率只要在数据库中直接统计即可。某一特定关联（“锤子和钉子”）在数据库中出现的频率称为*支持度*。比如在总共 1000 个事务中有 15 个事务同时包含了“锤子和钉子”，则此关联的支持度为 1.5%。非常低的支持度（比如 1 百万个事务中只有一个）可能意味着此关联不是很重要，或出现了错误数据（如，“男性和怀孕”）。

要找到有意义的规则，我们还要考察规则中项及其组合出现的*相对*频率。当已有 A 时，B 发生的概率是多少？也即概率论中的条件概率。回到我们的例子，也就是问“当一个人已经买了锤子，那他有多大的可能也会买钉子？”这个条件概率在数据挖掘中也称为*可信度*，

计算方法是求百分比：（A 与 B 同时出现的频率）/（A 出现的频率）。

让我们用一个例子更详细的解释这些概念：

总交易笔数（事务数）：1,000
包含“锤子”：50
包含“钉子”：80
包含“钳子”：20
包含“锤子”和“钉子”：15
包含“钳子”和“钉子”：10
包含“锤子”和“钳子”：10
包含“锤子”、“钳子”和“钉子”：5

则可以计算出：

“锤子和钉子”的支持度=1.5%（15/1,000）
“锤子、钉子和钳子”的支持度=0.5%（5/1,000）
“锤子==>钉子”的可信度=30%（15/50）
“钉子==>锤子”的可信度=19%（15/80）
“锤子和钉子==>钳子”的可信度=33%（5/15）
“钳子==>锤子和钉子”的可信度=25%（5/20）

我们可以看到买锤子的人也买钉子的可能性（30%）高于买钉子的人要买锤子的可能性（19%）。锤子和钉子关联的支持度已经足够高了，意味着这是一条有意义的关联规则。

改善度（*lift*）是另外一个描述规则价值的数值。改善度越高 A 的出现对 B 出现的可能性影响越大。改善度是一个比值：（A==>B 的可信度）/（B 出现的频率）。如：

“锤子==>钉子”的改善度：3.75（30% / 8%）
“锤子和钉子==>钳子”的改善度：16.5（33% / 2%）

关联规则算法的另一个重要的性质是指定项的概念层次。比如在我们讨论的锤子和钉子的例子中没有涉及产品的品牌和型号。这一点很重要，如在“金属制品-->五金工具-->钉子-->5 号钉子-->XX 厂的 5 号钉子”的概念层次上，基于不同的目的，你可能需要选择不同的层次。

注意数据挖掘得到的关联规则或序列模式并不是真正的规则，他只是对数据库中数据之间相关性的一种描述。还没有其他数据来验证得到的规则的正确性，也不能保证利用过去的数据得到的规律在未来新的情况下仍有效。

有时很难决定能利用你发现的关联规则做些什么。比如，在超市货架的摆放策略上，按照发现的关联规则把相关性很强的物品放在一起，反而可能会使整个超市的销售量下降—顾客如果可以很容易的找到他要买的商品，他就不会再买那些本来不在他的购买计划上的商品。总之，在采取任何行动之前一定要经过分析和实验，即使它是利用数据挖掘得到的知识。

有些软件产品用图形的方式显示项之间的相关性。如图 3 所示，每个圆圈代表一个项或一个事件，线代表他们间的关系，线越粗表示相关性越强，这样对软件的使用者来说就很直观。

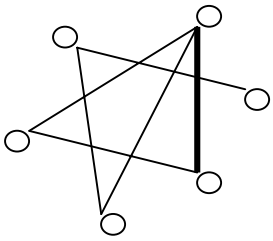

图 3：连接图

预言型数据挖掘

你需要选择的

数据挖掘的目的是生成可以据其所示的含义采取行动的知识，也就是建立一个现实世界的模型。建立这个模型可能需要各种各样的源数据，包括交易记录、顾客历史数据、人口统计信息、进程控制数据、和市场相关的外部数据等，比如：信用卡公司提供的数据、天气数据等。模型是模式和数据间相关性的形式化描述。

为了防止混淆，我们把数据挖掘概念划分为几个层次

- ✧ 商业目标
- ✧ 预言的种类
- ✧ 模型的类型
- ✧ 算法
- ✧ 产品

最高层是**商业目标**：数据挖掘的最终目的是什么？比如：希望用数据挖掘技术留住你的有价值的客户，你可能先要建立一个模型来预测每个客户所能带来的利润，然后再建立一个模型来确定哪些客户可能会离开。充分了解你所在企业的需求和目标有助于你建立这样的目标。

下一步是决定最合适的**预言的种类**：（1）分类：预测一个特定的客户或事件属于哪一类；（2）回归（regression）：预测一个变量的值（如果此变量随事件变化，可成为时间序列预测）。在上面的例子中你可以用回归来预测利润的大小，用分类预测哪些客户会离开。后面我们会详细讨论。

现在你可以选择**模型的类型**：用神经网络来做回归，决策树做分类，还是用统计模型，如：逻辑回归，偏差分析，普通线性模型等。下一章我们要详细讨论这些模型。

每种模型都可以用不同的**算法**来实现，比如，可以用回馈函数或 radial basis 函数来建立神经网络；决策树有 CART，C5.0，QUEST，CHAID 等。

在选择数据挖掘**软件产品**时，要注意这些软件所采用的算法虽然名称可能完全一样，但他们的实现方法通常都是不一样的。这些对算法的不同实现影响了软件对内存、硬盘的需求的不同，和性能上的差异。

大部分的商业目标都可以用各种不同的模型及相异的算法来解决。通常在你还没有试过任何数据挖掘算法之前，很难决定那种对你来说是最好的。

一些术语

在预言模型中，把我们要预测的值或所属类别称为响应变量、依赖变量或目标变量；用于预测的输入变量是预测变量或独立变量。

一些预言模型是通过那些已知目标变量值的历史数据训练出来的。这种训练有时也称为带指导的学习，因为是通过给出一些已知答案的问题（已知结果的数据）来让他“学习”。相对应的，还有不带指导的学习，如上面提到的描述型数据挖掘（在运行之前，算法对数据一无所知）。

分类

分类要解决的问题是为一个事件或对象归类。在使用上，既可以用此模型分析已有的数

据，也可以用它来预测未来的数据。例如，用分类来预测哪些客户最倾向于对直接邮件推销做出回应，又有哪些客户可能会换他的手机服务提供商，或在医疗领域当遇到一个病例时用分类来判断一下从哪些药品着手比较好。

数据挖掘算法的工作方法是通过分析已知分类信息的历史数据总结出一个预测模型。这里用于建立模型的数据称为训练集，通常是已经掌握的历史数据。如，已经不再接受服务的用户，你很可能还保存了他们在接受服务时的历史记录。训练集也可以是通过实际的实验得到的数据。比如你从包含公司所有顾客的数据库中取出一部分数据做实验，向他们发送介绍新产品的推销信，然后收集对此做出回应的客户名单，然后你就可以用这些推销回应记录建立一个预测哪些用户会对新产品感兴趣的模型，最后把这个模型应用到公司的所有客户上。

回归

回归是通过具有已知值的变量来预测其他变量的值。在最简单的情况下，回归采用的是象线性回归这样的标准统计技术。但在大多数现实世界中的问题是不能用简单的线性回归所能预测的。如商品的销售量、股票价格、产品合格率等，很难找到简单有效的方法来预测，因为要描述这些事件的变化所需的变量以上百计，且这些变量本身往往都是非线性的。为此人们又发明了许多新的手段来试图解决这个问题，如逻辑回归、决策树、神经网络等。

一般同一个模型既可用于回归也可用于分类。如 CART 决策树算法既可以用于建立分类树，也可建立回归树。神经网络也一样。

时间序列

时间序列是用变量过去的值来预测未来的值。与回归一样，他也是用已知的值来预测未来的值，只不过这些值的区别是变量所处时间的不同。时间序列采用的方法一般是在连续的时间流中截取一个时间窗口（一个时间段），窗口内的数据作为一个数据单元，然后让这个时间窗口在时间流上滑动，以获得建立模型所需要的训练集。比如你可以用前六天的数据来预测第 7 天的值，这样就建立了一个区间大小为 7 的窗口。

数据挖掘模型和算法

现在我们来讨论数据挖掘算法中涌到的各种类型的模型和算法。大多数数据挖掘产品使用的算法都是在计算机科学或统计数学杂志上发表过的成熟算法，所不同的只是算法的实现和对性能的优化。当然也有一些公司采用的是自己研发的未公开的算法，效果也不错。

我们下面将要介绍的模型和算法都是数据挖掘中最常见的和应用最广泛的，在计算机科学、统计数学、和人工智能领域的科学家们已经在研究和改进这些算法方面作了大量的工作。几乎所有的数据挖掘技术都可称为是数据驱动的，而不是用户驱动的，也就是说用户在使用这些算法时，只要给出数据，不用告诉算法程序怎么做和期待得到什么结果，一切都是算法自身从给定的数据中自己找出来。

应注意的是大部分算法都不是专为解决某个问题而特制的，算法之间也并不互相排斥。不能说一个问题一定要采用某种算法，别的就不行。一般来说并不存在所谓的最好的算法，在最终决定选取那种模型或算法之前，你可能各种模型都试一下，然后再选取一个较好的，或只试了一个就已经满足了你对准确度的要求。

神经网络

神经网络近来越来越受到人们的关注，因为它为解决大复杂度问题提供了一种相对来说比较有效的简单方法。神经网络可以很容易的解决具有上百个参数的问题（当然实际生物体中存在的神经网络要比我们这里所说的程序模拟的神经网络要复杂的多）。神经网络常用于两类问题：分类和回归。

在结构上，可以把一个神经网络划分为输入层、输出层和隐含层（见图 4）。输入层的每个节点对应一个个的预测变量。输出层的节点对应目标变量，可有多个。在输入层和输出层之间是隐含层（对神经网络使用者来说不可见），隐含层的层数和每层节点的个数决定了神经网络的复杂度。

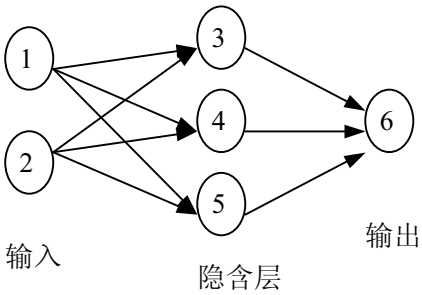

图 4：一个神经元网络

除了输入层的节点，神经网络的每个节点都与很多它前面的节点（称为此节点的输入节点）连接在一起，每个连接对应一个权重 W_{xy}，此节点的值就是通过它所有输入节点的值与对应连接权重乘积的和作为一个函数的输入而得到，我们把这个函数称为活动函数或挤压函数。如图 5 中节点 4 输出到节点 6 的值可通过如下计算得到：

W_{14}*节点 1 的值+W_{24}*节点 2 的值

神经网络的每个节点都可表示成预测变量（节点 1，2）的值或值的组合（节点 3-6）。注意节点 6 的值已经不再是节点 1、2 的线性组合，因为数据在隐含层中传递时使用了活动函数。实际上如果没有活动函数的话，神经元网络就等价于一个线性回归函数，如果此活动

函数是某种特定的非线性函数，那神经网络又等价于逻辑回归。

调整节点间连接的权重就是在建立（也称训练）神经网络时要做的工作。最早的也是最基本的权重调整方法是错误回馈法，现在较新的有变化坡度法、类牛顿法、Levenberg-Marquardt 法、和遗传算法等。无论采用那种训练方法，都需要有一些参数来控制训练的过程，如防止训练过度和控制训练的速度。

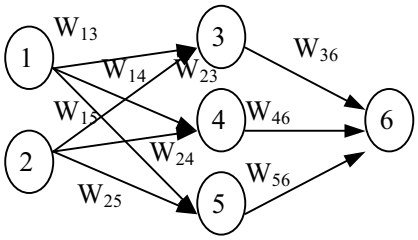

图 5：带权重 W_{xy} 的神经元网络

决定神经网络拓扑结构（或体系结构）的是隐含层及其所含节点的个数，以及节点之间的连接方式。要从头开始设计一个神经网络，必须要决定隐含层和节点的数目，活动函数的形式，以及对权重做那些限制等，当然如果采用成熟软件工具的话，他会帮你决定这些事情。

在诸多类型的神经网络中，最常用的是前向传播式神经网络，也就是我们前面图示中所描绘的那种。我们下面详细讨论一下，为讨论方便假定只含有一层隐含节点。

可以认为错误回馈式训练法是变化坡度法的简化，其过程如下：

前向传播：数据从输入到输出的过程是一个从前向后的传播过程，后一节点的值通过它前面相连的节点传过来，然后把值按照各个连接权重的大小加权输入活动函数再得到新的值，进一步传播到下一个节点。

回馈：当节点的输出值与我们预期的值不同，也就是发生错误时，神经网络就要“学习”（从错误中学习）。我们可以把节点间连接的权重看成后一节点对前一节点的“信任” 程度（他自己向下一节点的输出更容易受他前面哪个节点输入的影响）。学习的方法是采用惩罚的方法，过程如下：如果一节点输出发生错误，那么他看他的错误是受哪个（些）输入节点的影响而造成的，是不是他最信任的节点（权重最高的节点）陷害了他（使他出错），如果是则要降低对他的信任值（降低权重），惩罚他们，同时升高那些做出正确建议节点的信任值。对那些收到惩罚的节点来说，他也需要用同样的方法来进一步惩罚它前面的节点。就这样把惩罚一步步向前传播直到输入节点为止。

对训练集中的每一条记录都要重复这个步骤，用前向传播得到输出值，如果发生错误，则用回馈法进行学习。当把训练集中的每一条记录都运行过一遍之后，我们称完成一个*训练周期*。要完成神经网络的训练可能需要很多个训练周期，经常是几百个。训练完成之后得到的神经网络就是在通过训练集发现的模型，描述了训练集中响应变量受预测变量影响的变化规律。

由于神经网络隐含层中的可变参数太多，如果训练时间足够长的话，神经网络很可能把训练集的所有细节信息都“记”下来，而不是建立一个忽略细节只具有规律性的模型，我们称这种情况为训练过度。显然这种“模型”对训练集会有很高的准确率，而一旦离开训练集应用到其他数据，很可能准确度急剧下降。为了防止这种训练过度的情况，我们必须知道在什么时候要停止训练。在有些软件实现中会在训练的同时用一个测试集来计算神经网络在此测试集上的正确率，一旦这个正确率不再升高甚至开始下降时，那么就认为现在神经网络已经达到做好的状态了可以停止训练。

图 6 中的曲线可以帮我们理解为什么利用测试集能防止训练过度的出现。在图中可以看

到训练集和测试集的错误率在一开始都随着训练周期的增加不断降低，而测试集的错误率在达到一个谷底后反而开始上升，我们认为这个开始上升的时刻就是应该停止训练的时刻。

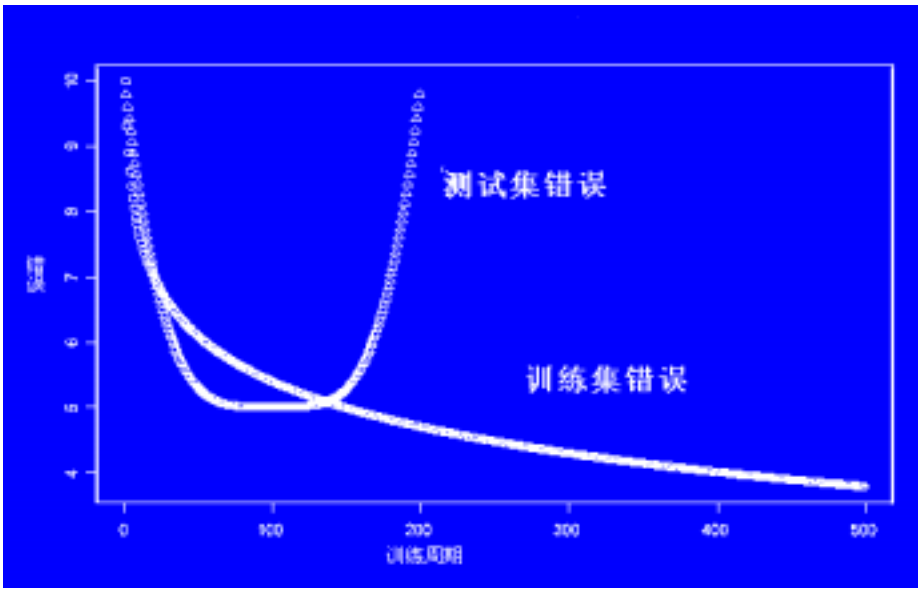

图 6：神经网络在训练周期增加时准确度的变化情况

神经元网络和统计方法在本质上有很多差别。神经网络的参数可以比统计方法多很多。如图 4 中就有 13 个参数（9 个权重和 4 个限制条件）。由于参数如此之多，参数通过各种各样的组合方式来影响输出结果，以至于很难对一个神经网络表示的模型做出直观的解释。实际上神经网络也正是当作“黑盒”来用的，不用去管“盒子”里面是什么，只管用就行了。在大部分情况下，这种限制条件是可以接受的。比如银行可能需要一个笔迹识别软件，但他没必要知道为什么这些线条组合在一起就是一个人的签名，而另外一个相似的则不是。在很多复杂度很高的问题如化学试验、机器人、金融市场的模拟、和语言图像的识别，等领域神经网络都取得了很好的效果。

神经网络的另一个优点是很容易在并行计算机上实现，可以把他的节点分配到不同的 CPU 上并行计算。

在使用神经网络时有几点需要注意：第一，神经网络很难解释，目前还没有能对神经网络做出显而易见解释的方法学。

第二，神经网络会学习过度，在训练神经网络时一定要恰当的使用一些能严格衡量神经网络的方法，如前面提到的测试集方法和交叉验证法等。这主要是由于神经网络太灵活、可变参数太多，如果给足够的时间，他几乎可以“记住”任何事情。

第三，除非问题非常简单，训练一个神经网络可能需要相当可观的时间才能完成。当然，一旦神经网络建立好了，在用它做预测时运行时还是很快得。

第四，建立神经网络需要做的数据准备工作量很大。一个很有误导性的神话就是不管用什么数据神经网络都能很好的工作并做出准确的预测。这是不确切的，要想得到准确度高的模型必须认真的进行数据清洗、整理、转换、选择等工作，对任何数据挖掘技术都是这样，神经网络尤其注重这一点。比如神经网络要求所有的输入变量都必须是 0-1（或-1 -- +1）之间的实数，因此像“地区”之类文本数据必须先做必要的处理之后才能用作神经网络的输入。

决策树

决策树提供了一种展示类似在什么条件下会得到什么值这类规则的方法。比如，在贷款申请中，要对申请的风险大小做出判断，图 7 是为了解决这个问题而建立的一棵决策树，从中我们可以看到决策树的基本组成部分：*决策节点*、*分支*和*叶子*。

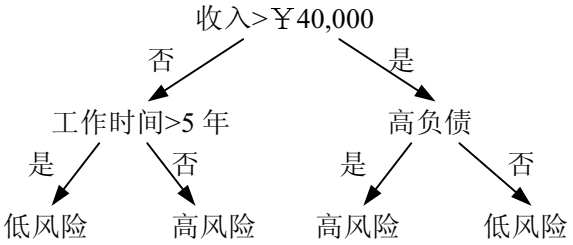

图 7：一棵简单的决策树

决策树中最上面的节点称为*根节点*，是整个决策树的开始。本例中根节点是“收入>￥40,000”，对此问题的不同回答产生了“是”和“否”两个分支。

决策树的每个节点子节点的个数与决策树在用的算法有关。如 CART 算法得到的决策树每个节点有两个分支，这种树称为*二叉树*。允许节点含有多于两个子节点的树称为*多叉树*。

每个分支要么是一个新的决策节点，要么是树的结尾，称为*叶子*。在沿着决策树从上到下遍历的过程中，在每个节点都会遇到一个问题，对每个节点上问题的不同回答导致不同的分支，最后会到达一个叶子节点。这个过程就是利用决策树进行分类的过程，利用几个变量（每个变量对应一个问题）来判断所属的类别（最后每个叶子会对应一个类别）。

假如负责借贷的银行官员利用上面这棵决策树来决定支持哪些贷款和拒绝哪些贷款，那么他就可以用贷款申请表来运行这棵决策树，用决策树来判断风险的大小。“年收入>￥40,00”和“高负债”的用户被认为是“高风险”，同时“收入<￥40,000”但“工作时间>5 年”的申请，则被认为“低风险”而建议贷款给他/她。

数据挖掘中决策树是一种经常要用到的技术，可以用于分析数据，同样也可以用来作预测（就像上面的银行官员用他来预测贷款风险）。常用的算法有 CHAID、 CART、 Quest 和 C5.0。

建立决策树的过程，即树的生长过程是不断的把数据进行切分的过程，每次切分对应一个问题，也对应着一个节点。对每个切分都要求分成的组之间的“差异”最大。

各种决策树算法之间的主要区别就是对这个“差异”衡量方式的区别。对具体衡量方式算法的讨论超出了本文的范围，在此我们只需要把切分看成是把一组数据分成几份，份与份之间尽量不同，而同一份内的数据尽量相同。这个切分的过程也可称为数据的“纯化”。看我们的例子，包含两个类别--低风险和高风险。如果经过一次切分后得到的分组，每个分组中的数据都属于同一个类别，显然达到这样效果的切分方法就是我们所追求的。

到现在为止我们所讨论的例子都是非常简单的，树也容易理解，当然实际中应用的决策树可能非常复杂。假定我们利用历史数据建立了一个包含几百个属性、输出的类有十几种的决策树，这样的一棵树对人来说可能太复杂了，但每一条从根结点到叶子节点的路径所描述的含义仍然是可以理解的。决策树的这种易理解性对数据挖掘的使用者来说是一个显著的优点。

然而决策树的这种明确性可能带来误导。比如，决策树每个节点对应分割的定义都是非常明确毫不含糊的，但在实际生活中这种明确可能带来麻烦（凭什么说年收入￥40,001 的人具有较小的信用风险而￥40,000 的人就没有）。

建立一颗决策树可能只要对数据库进行几遍扫描之后就能完成，这也意味着需要的计算资源较少，而且可以很容易的处理包含很多预测变量的情况，因此决策树模型可以建立得很快，并适合应用到大量的数据上。

对最终要拿给人看的决策树来说，在建立过程中让其生长的太“枝繁叶茂”是没有必要的，这样既降低了树的可理解性和可用性，同时也使决策树本身对历史数据的依赖性增大，也就是说这是这棵决策树对此历史数据可能非常准确，一旦应用到新的数据时准确性却急剧下降，我们称这种情况为*训练过度*。为了使得到的决策树所蕴含的规则具有普遍意义，必须防止训练过度，同时也减少了训练的时间。因此我们需要有一种方法能让我们在适当的时候停止树的生长。常用的方法是设定决策树的最大高度（层数）来限制树的生长。还有一种方法是设定每个节点必须包含的最少记录数，当节点中记录的个数小于这个数值时就停止分割。

与设置停止增长条件相对应的是在树建立好之后对其进行修剪。先允许树尽量生长，然后再把树修剪到较小的尺寸，当然在修剪的同时要求尽量保持决策树的准确度尽量不要下降太多。

对决策树常见的批评是说其在为一个节点选择怎样进行分割时使用“贪心”算法。此种算法在决定当前这个分割时根本不考虑此次选择会对将来的分割造成什么样的影响。换句话说，所有的分割都是顺序完成的，一个节点完成分割之后不可能以后再有机会回过头来再考察此次分割的合理性，每次分割都是依赖于他前面的分割方法，也就是说决策树中所有的分割都受根结点的第一次分割的影响，只要第一次分割有一点点不同，那么由此得到的整个决策树就会完全不同。那么是否在选择一个节点的分割的同时向后考虑两层甚至更多的方法，会具有更好的结果呢？目前我们知道的还不是很清楚，但至少这种方法使建立决策树的计算量成倍的增长，因此现在还没有哪个产品使用这种方法。

而且，通常的分割算法在决定怎么在一个节点进行分割时，都只考察一个预测变量，即节点用于分割的问题只与一个变量有关。这样生成的决策树在有些本应很明确的情况下可能变得复杂而且意义含混，为此目前新提出的一些算法开始在一个节点同时用多个变量来决定分割的方法。比如以前的决策树中可能只能出现类似“收入<￥35,000”的判断，现在则可以用“收入<(0.35*抵押)”或“收入>￥35,000 或抵押<150,000”这样的问题。

决策树很擅长处理非数值型数据，这与神经网络只能处理数值型数据比起来，就免去了很多数据预处理工作。

甚至有些决策树算法专为处理非数值型数据而设计，因此当采用此种方法建立决策树同时又要处理数值型数据时，反而要做把数值型数据映射到非数值型数据的预处理。

数据挖掘过程

在实施数据挖掘之前，先制定采取什么样的步骤，每一步都做什么，达到什么样的目标是必要的，有了好的计划才能保证数据挖掘有条不紊的实施并取得成功。很多软件供应商和数据挖掘顾问公司投提供了一些数据挖掘过程模型，来指导他们的用户一步步的进行数据挖掘工作。比如 SPSS 的 5A--评估（Assess），访问（Access），分析（Analyze），行动（Act），自动化（Automate），和 SAS 的 SEMMA--采样（Sample），探索（Explore），修正（Modify），建模（Model），评估（Assess）。

最近，一些软件供应商和用户组织成立了行业协会，包括 NCR Systems Engineering Copenhagen（丹麦）Daimler-Benz AG（德国）SPSS/Internal Solutions Ltd.（英国），和 OHRA Verzekeringen en Bank Grep B.V（荷兰）。这个组织的目的就是建立*跨行业数据挖掘过程标准*（CRISP-DM），在 1999 年 9 月的时候 CRISP-DM 仍在建立之中。

我们下面详细讨论 Two Crows 公司的数据挖掘过程模型，他与正在建立的 CRISP-DM 有许多相似之处。

数据挖掘过程模型

虽然我们把各个步骤按顺序排列，但要注意数据挖掘过程并不是线性的—要取得好的结果就要不断反复重复这些步骤。比如在“分析数据”时你可能觉得在“建立数据挖掘数据库”时作的不够好，要往里面添加一些新的数据。

这些基本数据挖掘步骤包括：

1. 定义商业问题
2. 建立数据挖掘模型
3. 分析数据
4. 准备数据
5. 建立模型
6. 评价模型
7. 实施

1. **定义商业问题。**在开始知识发现之前最先的同时也是最重要的要求就是了解的你的数据和业务问题。如果事先没有这种了解，没有任何算法，不管他有多么复杂玄妙，能够为你提供有价值的结果，即使有也难以使人信赖他。缺少了这些背景知识，你就没办法明确定义要解决的问题，不能为挖掘准备数据，也很难正确的解释得到的结果。要想充分发挥数据挖掘的价值，必须要对你的目标有一个清晰明确的定义，即决定到底想干什么。比如你说你想提高直接邮件推销的用户回应时，你想做的可能是“提高用户响应率”，也可能是“提高一次用户回应的价值”，要解决这两个问题而建立的模型几乎是完全不同的，你必须做出决定。有效的问题定义还应该包含一个对你的知识发现项目得到结果进行衡量的标准。当然还应该有整个项目预算和理性的解释。
2. **建立数据挖掘库。**连同下面的两个步骤，这三步构成了数据预处理的核心。这三步和在一起比其他所有的步骤加在一起所花得时间和精力还多。一旦你从数据挖掘的结果中学到一些什么之后，你很可能要修改数据以得到更好得结果，因此就需要把数据准备和数据挖掘不断的反复进行。*数据准备工作大概要花去整个数据挖掘项目的 50%-90%的时间和精力。*

应该把要挖掘的数据都收集到一个数据库中。注意这并不是说一定要使用一个数据库管理系统。根据要挖掘的数据量的大小、数据的复杂程度、使用方式的不同，有时一个简单的平面文件或电子表格就足够了。

一般来说，直接在公司的数据仓库上进行数据挖掘是不合适的。你最好建立一个独立的数据集。数据挖掘会使你成为数据仓库非常活跃的用户，这可能会带来一些资源申请上的问题。你需要经常把许多表连接在一起，访问数据仓库的细节数据。一个简单的试验在数据仓库内都要很多步才能完成。

大部分情况下你肯定需要修改要挖掘的数据。而且还会遇到把企业外部的数据拿到数据仓库内和在原有的表中增加新的字段的情况。其他的数据挖掘用户可能也要对数据仓库进行与您相似或完全不同的修改。而对数据仓库管理员来说，这恐怕是他最不愿意遇到的事情。

需要建立独立的数据挖掘库的另一个理由是，数据仓库可能不支持你要对数据进行各种复杂分析所需的数据结构。这包括对数据进行统计查询，多维分析，和各种复杂的图表和可视化。

最后，你可能希望把这些要挖掘的数据存贮在与公司的数据仓库在物理设计上不同的 DBMS 上。人们越来越倾向于使用 DBMS 本身很好的支持数据挖掘的数据库程序，这样能使数据挖掘工作进行的更容易一些。当然如果你的数据仓库允许你建立一个在逻辑上独立的数据库并且在计算资源上也足够的话，那么在他上面进行数据挖掘也是可以的。

可以把建立数据挖掘库分成下面几个部分：

a. 数据收集
b. 数据描述
c. 选择
d. 数据质量评估和数据清理
e. 合并与整合
f. 构建元数据
g. 加载数据挖掘库
h. 维护数据挖掘库

注意这些步骤并不需要一定要按步骤执行，而应该按需要进行。比如你可能在收集数据时就开始构建元数据，并随着工作的进行不断的对其进行修改。在数据整合和数据质量评估过程中了解到得东西也有可能是你修改最初的数据选择。

a. **数据收集**。确定要挖掘的数据源。可能一些外部的数据也是必须的，需要在公共数据库中获取（人口统计或天气数据）或向数据拥有者购买（比如信用卡使用数据）。

   用以一个*数据搜集报告*把所需的各种不同的数据源的属性列出来。此报告至少应包含如下的内容：

- ✧ 数据源（内部数据库或外部提供者）
- ✧ 拥有者
- ✧ 负责维护此数据的人/组织
- ✧ DBA
- ✧ 费用（如果需要购买）
- ✧ 存储方式（如：Oracle 数据库、MSAM 文件等）
- ✧ 表、字段、记录的数目
- ✧ 字节数
- ✧ 物理存储方式（CD-ROM、磁带、服务器等）

- ✧ 安全需求
- ✧ 使用上的限制
- ✧ 隐私上的需求

注意一旦应用了在安全和隐私上有特殊限制的数据，那么你的数据挖掘库在安全和隐私上也就继承了同样的限制。比如许多欧洲的数据在隐私上的限制要比美国严格的多。

b. **数据描述。**描述每个文件和数据库表的内容。*数据描述报告*中应包含如下内容：

- ✧ 字段/列的数目
- ✧ 字段是空（缺值）的数目/百分比
- ✧ 字段的名字

对每个字段

- ✧ 数据类型
- ✧ 定义
- ✧ 描述
- ✧ 源
- ✧ 计量单位
- ✧ 所有不同值的个数
- ✧ 值的列表
- ✧ 值的范围
- ✧ 空值的百分比
- ✧ 收集信息（比如：怎么得到的，在哪，什么条件下）
- ✧ 时间频度（Timeframe）（每天，每周，还是每月）
- ✧ 特别时间数据（比如：每个周一或每个周三）
- ✧ 主键/外键关系

c. **选择。**接下来就要选择用于数据挖掘的数据（源数据的子集）。这与对数据进行采样和选择预测变量是不同的，这里只是粗略的把一些冗余或无关的数据除去，或由于资源的限制、费用的限制、数据使用的限制、和质量问题而必须做出的选择。

d. **数据质量评估和数据清理。**“龙生龙，凤生凤”对数据挖掘也是非常适用的，要想得到好的模型必须用好的数据。数据质量评估就是要确定数据的哪些性质会最终影响模型的质量。你不仅要保证数据值的正确性和一致性，还要保证这些值是按同样的方法记录的同一件事情。

由各种各样的数据质量问题。数据域中可能包含了不正确的值。比如，身份证号码被粗心的数据录入人员录入了年龄。即使每个单个域中包含的数据都是正确的，但这些域组合起来时可能就出现了错误的记录，如男性怀孕。有时域中的值为空。当从多个不同的源整合数据时一定要注意不同源之间数据的一致性。

缺值是一个非常有害的问题。如果只要有一个数据域缺值就把这个记录删除掉，那么最后可能得到一个很小的数据库，同时你得到的这个数据库很可能已经丢失了实际数据中蕴含的一些信息，因为你已经改变了原数据的组成。缺值这件事本身可能就是非常有意义的，比如也许只有富有的顾客才经常忽略“收入”这一项。你可以增加一个新的变量来标识这些缺值的记录，然后用它建立一个模型，然后与按其他方法建立的模型进行比较，看哪个准确度更高一些。

另一种方法是为缺失的值计算一个替代值。计算替代值的常用方法包括使用形式值（为名词变量），中间值（为可排序变量），平均值（为连

续变量）。还有一个不是很常用的方法是按照数据库中值的分布规律为缺值的字段添值。比如如果数据库中包含 40%男性和 60%女性，那么在为那些性别子段缺失的记录添值时也按这个比例随机赋值。还有一种方法是为这个缺值的字段用数据挖掘技术建立一个预测模型，然后按照这个模型的预测结果添值，这种方法效果应该最好，当然也最花时间。

承认生活并不是十全十美是必要的，数据挖掘也是一样，你也不可能解决所有遇到的问题，只能做得尽量好一点。检查和修正数据质量问题是一项非常耗费时间和金钱的工作，对解决不了的问题，通常你只能采取折衷的办法，先用现有的数据建立一个模型，把问题放到将来去解决。

e. **合并与整合**。如果幸运的话，你需要的所有数据都在同一个数据库中（比如数据仓库），但大部分情况下这些数据是分布在不同的数据库中。数据可能分布在公司不同的部门、不同的应用中，甚至在公司外（人口数据）。

数据合并与整合把来自不同数据源的数据合并到同一个数据挖掘库中，并且要使那些本来存在冲突和不一致的数据一致化。不恰当的一致化是数据质量问题的一个主要来源。不同的数据库间在数据定义和使用上通常都存在巨大的差异。有些不一致问题是容易解决的，如同一客户的不同住址问题。然而有些则非常棘手。例如同一个客户有不同的名字—甚至更遭的情况—多个不同的客户标识号码。同一个名字被用在不同的数据项上（同名异意），或同一个数据项用了不同的名字（同意不同名）。还有单位上的不统一。比如人民币和港元之间不能不做换算就直接加减。

f. **构建元数据。**数据收集报告和数据描述报告是建立元数据的基础。本质上，这是一个描述数据库的数据库。他用于建立实际的数据库和为分析数据和建立模型提供辅助信息。

g. **加载数据挖掘库。**大多数情况下，用于挖掘的数据应该放到他自己独立的数据库中。如果数据量大并且复杂，那么他通常是一个 RDMS，反之只是一个简单的平面文件即可。经过前面所有的搜集、整理之后，现在开始把这些数据实际的加载过来了。依赖于所涉及的 DBMS 和操作系统，和数据库设计的复杂程度，有时这一步也可能变得很复杂，需要靠专家的帮助来完成。

h. **维护数据挖掘库。**挖掘库一旦建好，就需要对他进行维护。需要定期备份；监视他的性能；不时的增加存储空间或提高性能。对存放在 DBMS 内的复杂的挖掘库来说，维护他需要计算机专业人员来完成。

3. **分析数据。**请察看“描述型数据挖掘”以获得更详细的关于可视化、连结分析，及其他数据分析方法。分析的目的是找到对预测输出影响最大的数据字段，和决定是否需要定义导出字段。

如果数据集包含成百上千的字段，那么浏览分析这些数据将是一件非常耗时和累人的事情，这时你需要选择一个具有好的界面和功能强大的工具软件来协助你完成这些事情。

4. 准备数据。这是建立模型之前的最后一步数据准备工作。可以把此步骤划分成 4 个部分：

a. 选择变量
b. 选择记录
c. 创建新变量
d. 转换变量

a. **选择变量。**理想情况下，你可以选择你所有的全部变量，把他们输入到数据挖掘工具中，让他来帮你选择哪些是最好的预测变量。实际上这样做并不是很好，一方面是由于随着变量个数的增加，模型的建立时间也随之上升；另一方面盲目的把所有的变量都加进去会导致建立错误的模型。比如，建立预测模型的一个常见错误就是把一个依赖于目标变量的变量（由目标变量导出）作为预测变量，像用生日来“预测”年龄。

在原理上说，一些数据挖掘算法自动忽略不相关的变量、自动计算相关的（协）变量，在实际应用中完全依赖这些工具是不明智的，毕竟最了解你的数据的还是你自己。利用你的领域知识，你会做出大部分正确的选择。例如，用身份证号或人名做预测变量要么不会有任何用处，要么甚至降低了其他重要变量的影响力。

b. **选择记录。**与选择变量类似，你可能也想用你所有的数据行来建立模型，然而如果你的数据量确实非常巨大的话，要么要花费很长的时间来建立这个模型，要么买一台计算能力非常强大的机器。

因此，如果数据量特别大，进行抽样就是一个很好的主意。如果做的足够仔细，保证抽样是按真正的随机来进行的，采样对大部分商业问题来说都不会丢失信息。你可以用所有的数据建立一个模型；你还可以用采样的方法根据不同得采样方法建立几个模型，然后评价这几个模型选择一个最好的。我们认为后面这种方法得到的方法更准确、更健壮。

你可能选择数据中明显的异常数据删除掉。然而在某些情况下，这些看来异常的数据可能包含了你要建立模型的重要信息。基于你对问题的理解方式的不同，通常可以把这些异常忽略掉。比如可以把异常认为是人为的录入错误。

有时也需要向数据中添加一些新的数据（如，那些没有做出购买得客户）。

c. **创建新变量。**很多情况下需要从原始数据中衍生一些新的变量作为预测变量。比如，用负债占收入百分比来预测信用风险比直接用负债和收入做预测变量更准确一些，也更容易理解一些。很多变量如果组合起来（加、减、比率等）会比这些变量自身影响力更大。一些变量如果扩大它的范围也会成为一个非常好的预测变量，比如用一段时间内收入变化情况代替一个单一的收入数据。

d. **转换变量。**你所选择的算法和工具决定了都要对数据做哪些转换工作。如神经网络要求所有的变量都在 0-1 之间，因此在这些数据被提交到算法之前就必须先对不在[0,1]内的变量进行映射。同样一些决策树算法不接受数值型变量作为输入，在使用他们之前也要把这些数值映射到“高、中、低”等。当然你的转换方式也在一定程度上影响了模型的准确度。

5. **建立模型。**对建立模型来说要记住的最重要的事是它是一个反复的过程。你需要仔细考察不同的模型以判断哪个模型对你的商业问题最有用。你在寻找好的模型的过程中学到的的东西会启发你修改你的数据，甚至改变最初对问题的定义。

一旦决定了预测的类型之后（分类还是回归），就需要为这个预测选择模型的类型。可能是一棵决策树、神经网络、甚至传统的数学统计。选择什么样的模型决定了你需对数据做哪些预处理工作。如神经网络需要做数据转换，有些数据挖掘工具可能对输入数据的格式有特定的限制，等。一旦所有的数据准备好之后，就可以开始训练你的模型了。

为了保证得到的模型具有较好的精确度和健壮性，需要一个定义完善的训练—验证协议。有时也称此协议为带指导的学习。他的主要思想就是先用一部分数据建立模型，然后再用剩下的数据来测试和验证这个得到的模型。有时还有第三个数据

集，称为验证集，因为测试集可能受模型的特性的影响，这时需要一个独立的数据集来验证模型的准确性。

训练和测试数据挖掘模型需要把数据至少分成两个部分：一个用于模型训练，另一个用于模型测试。如果不使用不同的训练和测试集，那么模型的准确度就很难使人信服。用训练集把模型建立出来之后，就可以先在测试集数据上先试验一把，此模型在测试集上的预测准确度就是一个很好的指导数字，它说明如果将来与训练集和测试集类似的数据用此模型预测时，正确的百分比会有多大。这并不能保证模型的正确性，他只是说相似的数据用此模型会得出相似的结果。

简单验证。最基本的测试方法是被称为简单验证的方法。做法是从原始数据中拿出一定百分比的数据作为测试数据，这个百分比大概在 5%到 33%之间。注意在把数据库分成几部分时，一定要保证选择的随机性，这样才能使分开的各部分数据的性质是一致的。

先用数据库的主体把模型建立起来，然后用此模型来预测测试集中的数据。出现错误的预测与预测总数之间的比，称为*错误率*。正确的预测与总数的比，是*准确率*（准确率=1 - 错误率）。对回归模型来说，可以用方差来描述准确的程度。

在一次模型的建立过程中，即使这种最简单的验证就要执行几十次。例如在训练神经网络时，有时每一个训练周期就要求在测试集上运行一次，不断的训练测试，直到在测试集上的准确率不再提高为止。

交叉验证。如果数据不是很多（比如只有几千条），那么你可能承受不了再把一部分数据拿到一边不用，单用来做简单验证。交叉验证提供了一种让你使用全部数据的方法。首先把原始数据随机平分成两份，然后用一部分做训练集另一部分做测试集计算错误率，做完之后把两部分数据交换再计算一次，得到另一个错误率，最后再用所有的数据建立一个模型，把上面得到的两个错误率进行平均作为最后用所有数据建立的模型的错误率。

更通用的算法是 *n-维交叉验证*。先把数据随机分成不相交的 n 份。比如，如果把数据分成 10 份，先把第一份拿出来放在一边用作模型测试，把其他 9 份合在一起来建立模型，然后把这个用 90%的数据建立起来的模型用上面放在一边的第一份数据做测试。这个过程对每一份数据都重复进行一次，得到 10 个不同的错误率。最后把所有数据放在一起建立一个模型，模型的错误率为上面 10 个错误率的平均。

*自举法*是另一种评估模型错误率的技术。在数据量很小时尤其适用。与交叉验证一样模型是用所有的数据建立。

依据所得到的模型和你对模型的预期结果，你可能修改参数用同样的算法再建立新的模型，甚至采用其他的算法建立模型。别的方法说不定能提高预测的准确度。当然，没有一种算法或工具适应所有的数据，通常也很难在开始决定那种算法对你所面临的问题来说是最好的，因此很多情况下，需要建立用不同的方法（参数或算法）几个模型，从中选择最好的。

6. 评价和解释。

a. **模型验证。**模型建立好之后，必须评价他的结果、解释他的价值。记住从测试集中得到的准确率只对用于建立模型的数据有意义。在实际应用中，随着应用数据的不同，模型的准确率肯定会变化。更重要的是，准确度自身并不一定是选择最好模型的正确评价方法。你需要进一步了解错误的类型和由此带来的相关费用的多少。

无序矩阵。对分类问题来说，无序矩阵是理解结果的非常好的工具。如图 9 所示，无序矩阵把预测的结果与实际的情况进行对比。它不仅说明了模型预测的准确情况，也把模型究竟在哪里出了问题也展示了出来。下表是一个简单的无序矩阵，其中的列代表数据的实际分类情况，行是预测的结果。在这张表值中，可以看到此模型在总共 46 个 B 类数据中成功预测了 38 个，8 个出了问题：2 个预测成了 A，6 个成了 C。显然这比简单的说模型总体准确度是 82%要更详细的多（123 个成功预测除以总共 150）。

预测	实际		
	类别 A	类别 B	类别 C
类别 A	45	2	3
类别 B	10	38	2
类别 C	4	6	40

图 9：无序矩阵

在实际应用中，如果每种不同的预测错误所需付出的代价（金钱）也不同的话，那么代价最小的模型（而不一定是错误率最小的模型）就是我们所要选择的。例如，上面的无序矩阵中，如果每个准确的预测会带来￥10 的收益，错误的预测 A 要付出￥5 的代价，B 是￥10，C 是￥20，那么整个模型的纯价值是：

(123*￥10)-（5*￥5）-（12*￥10）-（10*￥20）=￥885

然而考察下面的无序矩阵（图 10），虽然准确度降低到 79%（118/150），但纯价值却升高了：

（118*￥10）-（22*￥5）-（7*￥10）-（3*￥20）=￥940

预测	实际		
	类别 A	类别 B	类别 C
类别 A	40	12	10
类别 B	6	38	1
类别 C	2	1	40

图 10：另一个无序矩阵

收益表（图 11）也是一种描述模型价值的方法。它显示了通过应用模型*响应*（如直接邮件推销）的变化情况。变化的比率称为 lift。例如，如果用随机抽取的方法选择 10%的客户响应率是 10%，而通过模型选取 10%的用户响应率是 30%，则 lift 值为 3。

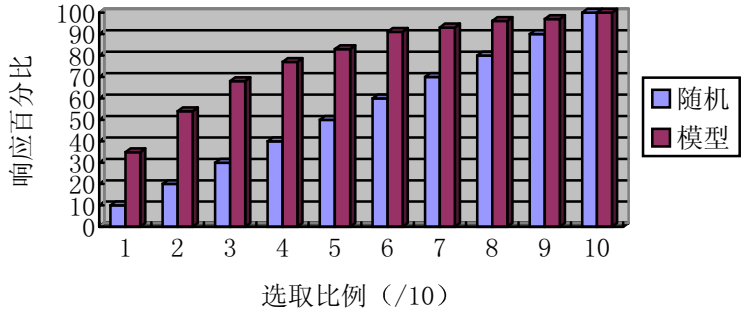

图11：收益表

模型解释的另一个重要组成部分是确定模型的价值。一个模型可能看起来很有意义，但要实施它的话很有可能花的钱比赚的钱多。图 12 是一个描述模型投资回报率（ROI）的图表（这里定义 ROI 为利润与开销的比值）。注意图中当选取比例超过 80%时，ROI 变成了负数，ROI 最高是在横坐标为 2 时。

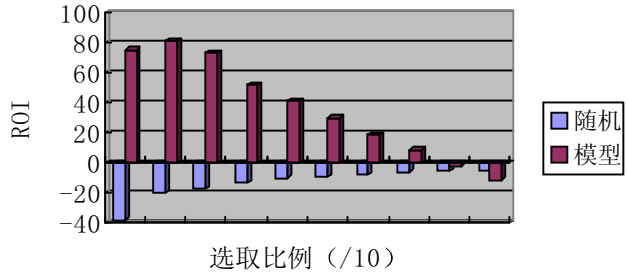

图12：ROI表

当然，也可以直接看利润的变化情况（利润为收入与花费的差值），如图 13 所示：

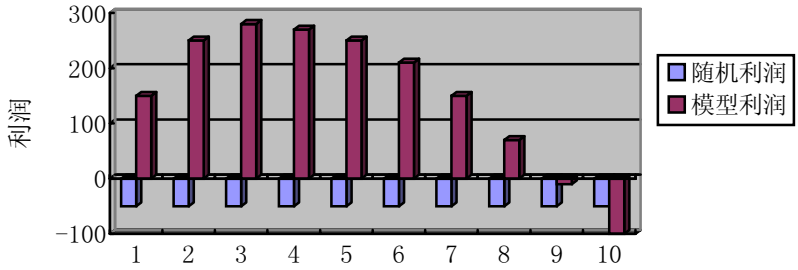

图13：利润表

注意到我们上面的例子中，最大 lift 在第一个 1/10 处（10%），最大 ROI 在第 2 个 1/10（20%），而最大利润在第 3、4 个 1/10 处。

理想情况下，应该按照利润表行事，但很多情况下利润表并不能很容易的计算出来。

b. **外部验证。**如前面指出的，无论我们用模拟的方法计算出来的模型的准确率有多高，都不能保证此模型在面对现实世界中真实的数据时能取得好的效果。经验证有效的模型并不一定是正确的模型。造成这一点的直接原因就是模型建立中隐含的各种假定。例如，在建立用户购买模式的模型时，可能没有考虑通货膨胀的影响，但实施模型时通货膨胀率突然由 3%增加为 17%，这显然会对人们的购买意向产生重大影响，因此再用原来的模型来预测客户购买情况必然会出现重大失误。

因此直接在现实世界中测试模型很重要。先在小范围内应用，取得测试数据，觉得满意之后再向大范围推广。

7. **实施。**模型建立并经验证之后，可以有两种主要的使用方法。第一种是提供给分析人员做参考，由他通过察看和分析这个模型之后提出行动方案建议。比如可以把模型检测到的聚集、模型中蕴含的规则、或表明模型效果的 ROI 图表拿给分析人员看。

另一种是把此模型应用到不同的数据集上。模型可以用来标示一个事例的类别，给一项申请打分等。还可以用模型在数据库中选择符合特定要求的记录，以用

OLAP 工具做进一步的分析。

通常情况下，模型是某个商业过程的组成部分，如风险分析，信用授权，或欺诈检测。在这些情况下，模型一般都合并到应用程序的内部。例如，在抵押贷款应用程序内部可能集成了一个预测模型，来向贷款官员提供一项贷款申请风险大小的建议。或在定购系统中，当预测到库存有可能降低到一个最低限度时自动发出购买订单。

数据挖掘模型通常一次只能处理一个事件或一个事务。每个事务的处理时间和事务到达的速度，决定了模型运行所需的计算能力，和是否要用并行技术来加快速度。比如，贷款申请程序可能 PC 机上就运行的很好，而用于信用卡欺诈的模型则需要在服务器上用并行算法才能应付每天的大量事务。

当提交一个复杂的应用时，数据挖掘可能只是整个产品的一小部分，虽然可能是最关键的一部分。例如，常常把数据挖掘得到的知识与领域专家的知识结合起来，然后应用到数据库中的数据。在欺诈检测系统中可能既包含了数据挖掘发现的规律，也有人们在实践中早已总结出的规律。

模型监控。在应用了模型之后，当然还要不断监控他的效果。即使你在开始使用这个模型之后觉得他非常成功，也不能放弃监控，因为事物在不断发展变化，很可能过一段时间之后，模型就不再起作用。销售人员都知道，人们的购买方式随着社会的发展而变化。因此随着使用时间的增加，要不断的对模型做重新测试，有时甚者需要重新建立模型。