第 8 章
机器学习和数据挖掘

机器学习（Machine Learning，ML）是一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、算法复杂度理论等多门学科。其专门研究计算机是怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构，使之不断改善自身的性能。

除了机器学习外，本章还将涉及另外一个领域——数据挖掘，它和机器学习有很大的交集。

机器学习和数据挖掘是两个非常难的领域，本章更多地从架构和应用角度去解读，理论知识则不进行重点阐述。

8.1 机器学习和数据挖掘的联系与区别

数据挖掘是从海量数据中获取有效的、新颖的、潜在有用的、最终可理解的模式的非平凡过程。数据挖掘中用到了大量的机器学习界提供的数据分析技术和数据库界提供的数据管理技术。从数据分析的角度来看，数据挖掘与机器学习有很多相似之处，但不同之处也十分明显，例如，数据挖掘并没有机器学习探索人的学习机制这一科学发现任务，数据挖掘中的数据分析是针对海量数据进行的，等等。从某种意义上说，机器学习的科学成分更重一些，而数据挖掘的技术成分更重一些。

学习能力是智能行为的一个非常重要的特征，不具有学习能力的系统很难称之为一个真正的智能系统，而机器学习则希望（计算机）系统能够利用经验来改善自身的性能，因此该领域一直是人工智能的核心研究领域之一。在计算机系统中，“经验”通常是以数据的形式存在的，因此，机器学习不仅涉及对人的认知学习过程的探索，还涉及对数据的分析处理。实际上，机器学习已经成为计算机数据分析技术的创新源头之一。由于几乎所有的学科都要面对数据分析任务，因此机器学习已经开始影响到计算机科学的众多领域，甚至影响到计算机科学之外的很多学科。机器学习是数据挖掘中的一种重要工具。然而数据挖掘不仅仅要研究、拓展、应用一些机器学习方法，还要通过许多非机器学习技术解决数据仓储、大规模数据、数据噪声等实践问题。机器学习的涉及面也很宽，常用在数据挖掘上的方法通常只是“从数据学习”。然而机器学习不仅仅可以用在数据挖掘上，一些机器学习的子领域甚至与数据挖掘关系不大，如增强学习与自动控制等。所以笔者认为，数据挖掘是从目的而言的，机器学习是从方法而言的，两个领域有相当大的交集，但不能等同。

8.2　典型的数据挖掘和机器学习过程

图 8.1 是一个典型的推荐类应用，需要找到“符合条件的”潜在人员。要从用户数据中得出这张列表，首先需要挖掘出客户特征，然后选择一个合适的模型来进行预测，最后从用户数据中得出结果。

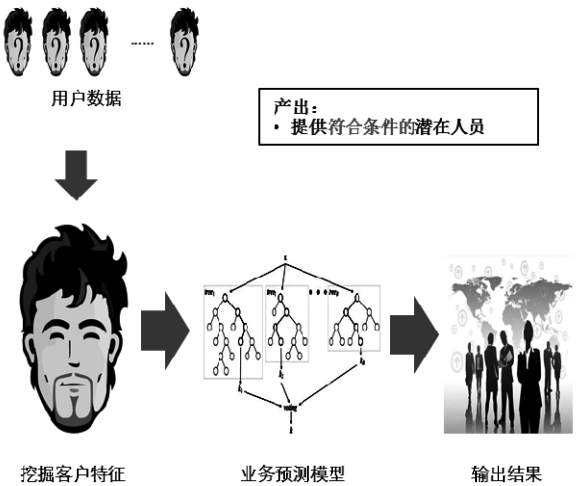

图 8.1

把上述例子中的用户列表获取过程进行细分，有如下几个部分（见图 8.2）。

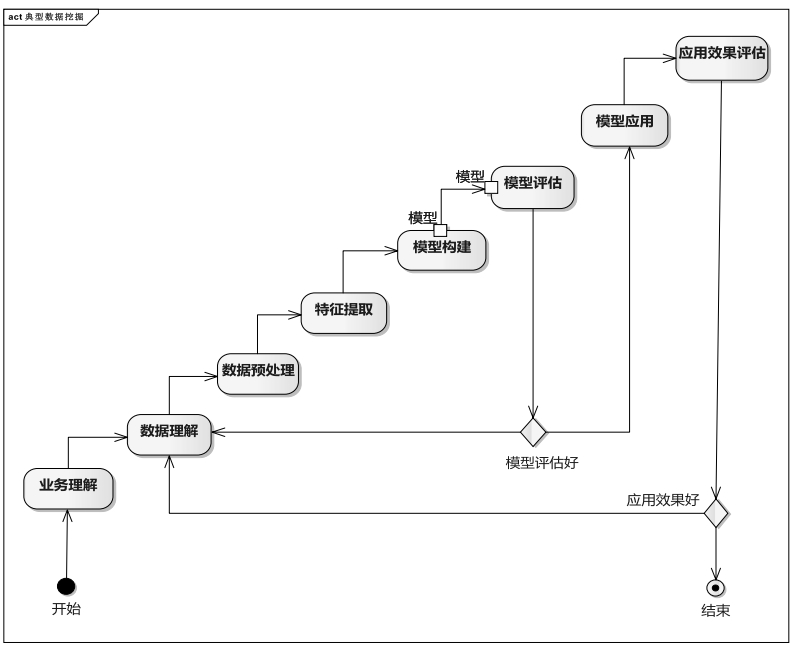

图 8.2

- 业务理解：理解业务本身，其本质是什么？是分类问题还是回归问题？数据怎么获取？应用哪些模型才能解决？
- 数据理解：获取数据之后，分析数据里面有什么内容、数据是否准确，为下一步的预处理做准备。
- 数据预处理：原始数据会有噪声，格式化也不好，所以为了保证预测的准确性，需要进行数据的预处理。
- 特征提取：特征提取是机器学习最重要、最耗时的一个阶段。
- 模型构建：使用适当的算法，获取预期准确的值。
- 模型评估：根据测试集来评估模型的准确度。
- 模型应用：将模型部署、应用到实际生产环境中。
- 应用效果评估：根据最终的业务，评估最终的应用效果。

整个过程会不断反复，模型也会不断调整，直至达到理想效果。

8.3 机器学习概览

机器学习的算法有很多，这里从两个方面进行介绍：一个是学习方式，另一个是算法类似性。

8.3.1 学习方式

根据数据类型的不同，对一个问题的建模可以有不同的方式。在机器学习或人工智能领域，人们首先会考虑算法的学习方式。在机器学习领域有如下几种主要的学习方式。

1．监督式学习

在监督式学习下，输入数据被称为“训练数据”，每组训练数据都有一个明确的标识或结果，如对防垃圾邮件系统中的“垃圾邮件”、“非垃圾邮件”，对手写数字识别中的“1”、“2”、“3”、“4”等。在建立预测模型的时候，监督式学习建立一个学习过程，将预测结果与“训练数据”的实际结果进行比较，不断地调整预测模型，直到模型的预测结果达到一个预期的准确率。监督式学习的常见应用场景包括分类问题和回归问题。常见算法有逻辑回归（Logistic Regression）和反向传递神经网络（Back Propagation Neural Network）。

2．非监督式学习

在非监督式学习下，数据并不被特别标识，学习模型是为了推断出数据的一些内在结构。常见的应用场景包括关联规则的学习及聚类等。常见算法包括 Apriori 算法和 K-Means 算法。

3．半监督式学习

在半监督式学习下，输入数据部分被标识，部分没有被标识。这种学习模型可以用来进行预测，但是模型首先需要学习数据的内在结构，以便合理地组织数据进行预测。其应用场景包括分类和回

归。常见算法包括一些对常用监督式学习算法的延伸。这些算法首先试图对未标识的数据进行建模，然后在此基础上对标识的数据进行预测，如图论推理算法（Graph Inference）或拉普拉斯支持向量机（Laplacian SVM）等。

4．强化学习

在强化学习下，输入数据作为对模型的反馈，不像监督模型那样，输入数据仅仅作为一种检查模型对错的方式。在强化学习下，输入数据直接反馈到模型，模型必须对此立刻做出调整。常见的应用场景包括动态系统及机器人控制等。常见算法包括 Q-Learning 及时间差学习（Temporal Difference Learning）等。

在企业数据应用的场景下，人们最常用的可能就是监督式学习和非监督式学习。在图像识别等领域，由于存在大量的非标识数据和少量的可标识数据，目前半监督式学习是一个很热门的话题。而强化学习更多地应用在机器人控制及其他需要进行系统控制的领域。

8.3.2　算法类似性

根据算法的功能和形式的类似性，可以对算法进行分类，如基于树的算法、基于神经网络的算法等。当然，机器学习的范围非常庞大，有些算法很难明确归到某一类。而对于有些分类来说，同一分类的算法可以针对不同类型的问题。这里，我们尽量把常用的算法按照最容易理解的方式进行分类。

1．回归算法

回归算法是试图采用对误差的衡量来探索变量之间的关系的一类算法。回归算法是统计机器学习的利器。常见的回归算法包括最小二乘法（Ordinary Least Square）、逻辑回归（Logistic Regression）、逐步式回归（Stepwise Regression）、多元自适应回归样条（Multivariate Adaptive Regression Splines）及本地散点平滑估计（Locally Estimated Scatterplot Smoothing）等。

2．基于实例的算法

基于实例的算法常常用来对决策问题建立模型，这样的模型常常先选取一批样本数据，然后根据某些近似性把新数据与样本数据进行比较，从而找到最佳的匹配。因此，基于实例的算法常常被称为“赢家通吃学习”或者“基于记忆的学习”。常见的算法包括 k-Nearest Neighbor（kNN）、学习矢量量化（Learning Vector Quantization，LVQ）及自组织映射算法（Self-Organizing Map，SOM）等。

3．正则化算法

正则化算法是其他算法（通常是回归算法）的延伸，根据算法的复杂度对算法进行调整。正则化算法通常对简单模型予以奖励，而对复杂算法予以惩罚。常见的算法包括 Ridge Regression、Least Absolute Shrinkage and Selection Operator（LASSO）及弹性网络（Elastic Net）等。

4．决策树算法

决策树算法根据数据的属性采用树状结构建立决策模型，常常用来解决分类和回归问题。常见的算法包括分类及回归树（Classification and Regression Tree，CART）、ID3（Iterative Dichotomiser 3）、C4.5、Chi-squared Automatic Interaction Detection（CHAID）、Decision Stump、随机森林（Random Forest）、多元自适应回归样条（MARS）及梯度推进机（Gradient Boosting Machine，GBM）等。

5．贝叶斯算法

贝叶斯算法是基于贝叶斯定理的一类算法，主要用来解决分类和回归问题。常见的算法包括朴素贝叶斯算法、平均单依赖估计（Averaged One-Dependence Estimators，AODE）及 Bayesian Belief Network（BBN）等。

6．基于核的算法

基于核的算法中最著名的莫过于支持向量机（SVM）。基于核的算法是把输入数据映射到一个高阶的向量空间，在这些高阶向量空间里，有些分类或者回归问题能够更容易地解决。常见的基于核的算法包括支持向量机（Support Vector Machine，SVM）、径向基函数（Radial Basis Function，RBF）及线性判别分析（Linear Discriminate Analysis，LDA）等。

7．聚类算法

聚类算法通常按照中心点或者分层的方式对输入数据进行归并。所有的聚类算法都试图找到数据的内在结构，以便按照最大的共同点将数据进行归类。常见的聚类算法包括 K-Means 算法及期望最大化算法（Expectation Maximization，EM）等。

8．关联规则学习

关联规则学习通过寻找最能够解释数据变量之间关系的规则，来找出大量多元数据集中有用的关联规则。常见的算法包括 Apriori 算法和 Eclat 算法等。

9．人工神经网络算法

人工神经网络算法模拟生物神经网络，是一类模式匹配算法，通常用于解决分类和回归问题。人工神经网络是机器学习的一个庞大的分支，有几百种不同的算法（深度学习就是其中的一类算法）。常见的人工神经网络算法包括感知器神经网络（Perceptron Neural Network）、反向传递（Back Propagation）、Hopfield 网络、自组织映射（Self-Organizing Map，SOM）及学习矢量量化（Learning Vector Quantization，LVQ）等。

10．深度学习算法

深度学习算法是对人工神经网络的发展。在计算能力变得日益廉价的今天，深度学习算法试图建立大得多也复杂得多的神经网络。很多深度学习算法是半监督式学习算法，用来处理存在少量未标识数据的大数据集。常见的深度学习算法包括受限波尔兹曼机（Restricted Boltzmann Machine，RBN）、Deep Belief Networks（DBN）、卷积网络（Convolutional Network）及堆栈式自动编码器（Stacked Auto-encoders）等。

11．降低维度算法

与聚类算法一样，降低维度算法试图分析数据的内在结构，不过降低维度算法通过非监督式学习，试图利用较少的信息来归纳或者解释数据。这类算法可以用于高维数据的可视化，或者用来简化数据以便监督式学习使用。常见的降低维度算法包括主成分分析（Principle Component Analysis，PCA）、偏最小二乘回归（Partial Least Square Regression，PLSR）、Sammon 映射、多维尺度（Multi-Dimensional Scaling，MDS）及投影追踪（Projection Pursuit）等。

12．集成算法

集成算法用一些相对较弱的学习模型独立地就同样的样本进行训练，然后把结果整合起来进行整体预测。集成算法的主要难点在于究竟集成哪些独立的、较弱的学习模型，以及如何把学习结果整合起来。这是一类非常强大的算法，同时也非常流行。常见的集成算法包括 Boosting、Bootstrapped Aggregation（Bagging）、AdaBoost、堆叠泛化（Stacked Generalization，Blending）、梯度推进机（Gradient Boosting Machine，GBM）及随机森林（Random Forest）等。

8.4　机器学习&数据挖掘应用案例

前面了解了机器学习和数据挖掘的基本概念，本节来看一下业界成熟的案例，让读者对机器学习和数据挖掘有一个直观的理解。

8.4.1　尿布和啤酒的故事

先来看一则有关数据挖掘的故事——“尿布与啤酒”。

总部位于美国阿肯色州的世界著名商业零售连锁企业沃尔玛拥有世界上最大的数据仓库系统。为了能够准确了解顾客在其门店的购买习惯，沃尔玛对其顾客的购物行为进行购物篮分析，想知道顾客经常一起购买的商品有哪些。沃尔玛数据仓库里集中了其各门店的详细原始交易数据，在这些原始交易数据的基础上，沃尔玛利用 NCR 数据挖掘工具对这些数据进行分析和挖掘。一个意外的发现是：跟尿布一起购买最多的商品竟然是啤酒！这是数据挖掘技术对历史数据进行分析的结果，反映了数据的内在规律。那么，这个结果符合现实情况吗？是否有利用价值？

于是，沃尔玛派出市场调查人员和分析师对这一数据挖掘结果进行调查分析，从而揭示出隐藏在“尿布与啤酒”背后的美国人的一种行为模式：在美国，一些年轻的父亲下班后经常要到超市去买婴儿尿布，而他们中有 30%～40%的人同时也为自己买一些啤酒。产生这一现象的原因是：美国的太太们常叮嘱她们的丈夫下班后为小孩买尿布，而丈夫们在买完尿布后又随手带回了他们喜欢的啤酒。

既然尿布与啤酒一起被购买的机会很多，于是沃尔玛就在其各家门店将尿布与啤酒摆放在一起，结果是尿布与啤酒的销售量双双增长。

8.4.2 决策树用于电信领域故障快速定位

电信领域比较常见的应用场景是决策树，利用决策树来进行故障定位。比如，用户投诉上网慢，其中就有很多种原因，有可能是网络的问题，也有可能是用户手机的问题，还有可能是用户自身感受的问题。怎样快速分析和定位出问题，给用户一个满意的答复？这就需要用到决策树。

图 8.3 就是一个典型的用户投诉上网慢的决策树的样例。

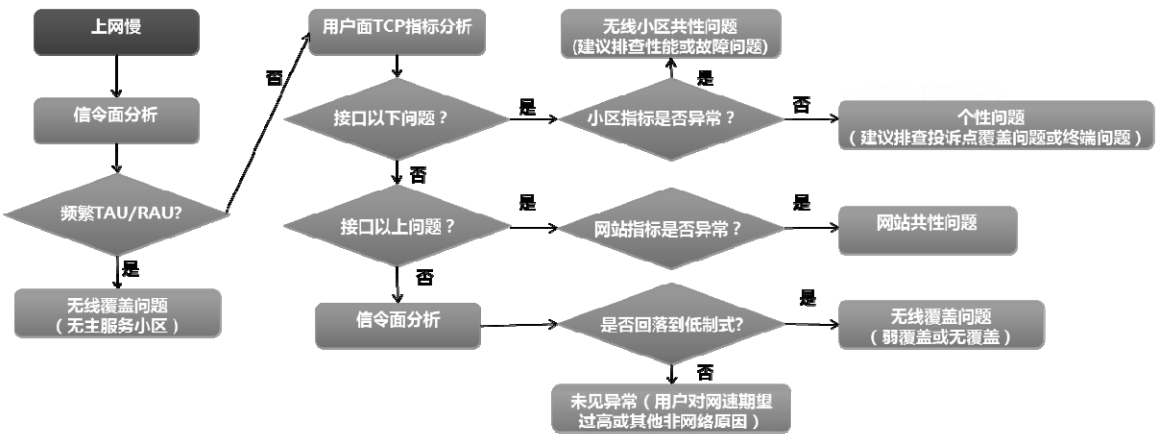

图 8.3

8.4.3 图像识别领域

1．小米面孔相册

这项功能的名字叫“面孔相册”，可以利用图像分析技术，自动地对云相册照片内容按照面孔进行分类整理。开启“面孔相册”功能后，可以自动识别、整理和分类云相册中的不同面孔。

“面孔相册”还支持手动调整分组、移出错误面孔、通过系统推荐确认面孔等功能，从而弥补机器识别的不足。

这项功能的背后其实使用的是深度学习技术，自动识别图片中的人脸，然后进行自动识别和分类。

2．支付宝扫脸支付

马云在 2015 CeBIT 展会开幕式上首次展示了蚂蚁金服的最新支付技术“Smile to Pay”（扫脸支付），惊艳全场。支付宝宣称，Face++ Financial 人脸识别技术在 LFW 国际公开测试集中达到 99.5% 的准确率，同时还能运用“交互式指令+连续性判定+3D 判定”技术。人脸识别技术基于神经网络，让计算机学习人的大脑，并通过“深度学习算法”大量训练，让它变得极为“聪明”，能够“认人”。实现人脸识别不需要用户自行提交照片，有资质的机构在需要进行人脸识别时，可以向全国公民身份证号码查询服务中心提出申请，将采集到的照片与该部门的权威照片库进行比对。

也就是说，用户在进行人脸识别时，只需打开手机或电脑的摄像头，对着自己的正脸进行拍摄

即可。在智能手机全面普及的今天，这个参与门槛低到可以忽略不计。

用户容易担心的隐私问题在人脸识别领域也能有效避免，因为照片来源权威，同时，一种特有的“脱敏”技术可以将照片模糊处理成肉眼无法识别而只有计算机才能识别的图像。

3．图片内容识别

前面两个案例介绍的都是图片识别，比图片识别更难的是图片语义的理解和提取，百度和 Google 都在进行这方面的研究。

百度的百度识图能够有效地处理特定物体的检测识别（如人脸、文字或商品）、通用图像的分类标注，如图 8.4 所示。

图 8.4

来自 Google 研究院的科学家发表了一篇博文，展示了 Google 在图形识别领域的最新研究进展。或许未来 Google 的图形识别引擎不仅能够识别出图片中的对象，还能够对整个场景进行简短而准确的描述，如图 8.5 所示。这种突破性的概念来自机器语言翻译方面的研究成果：通过一种递归神经网络（RNN）将一种语言的语句转换成向量表达，并采用第二种 RNN 将向量表达转换成目标语言的语句。

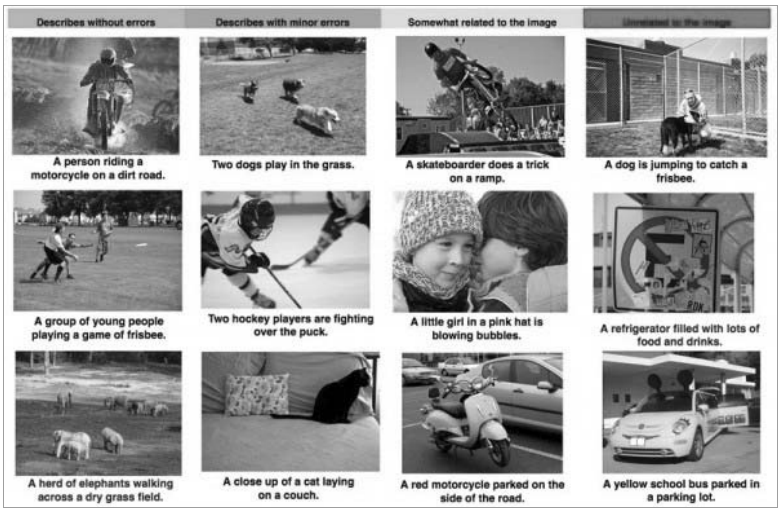

图 8.5

而 Google 将以上过程中的第一种 RNN 用深度卷积神经网络 CNN 替代，这种网络可以用来识别图像中的物体。通过这种方法可以实现将图像中的对象转换成语句，对图像场景进行描述。概念虽

然简单，但实现起来十分复杂，科学家表示目前实验产生的语句合理性不错，但距离完美仍有差距，这项研究目前仅处于早期阶段。图 8.6 展示了通过此方法识别图像对象并产生描述的过程。

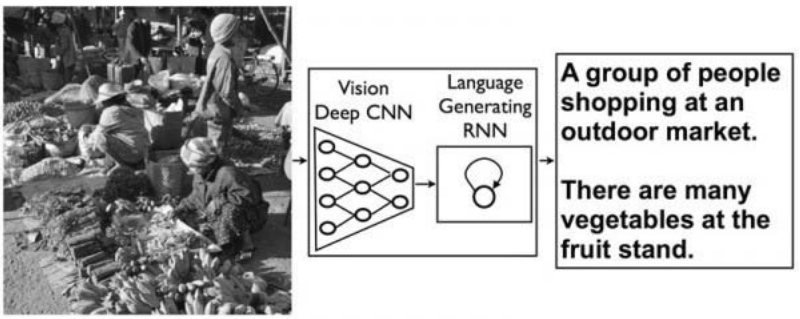

图 8.6

8.4.4 自然语言识别

自然语言识别一直是一个非常热门的领域，最有名的是苹果的 Siri，支持资源输入，调用手机自带的天气预报、日常安排、搜索资料等应用，还能够不断学习新的声音和语调，提供对话式的应答。

微软的 Skype Translator 可以实现中英文之间的实时语音翻译功能，将使得英文和中文普通话之间的实时语音对话成为现实。

Skype Translator 的运作机制如图 8.7 所示。

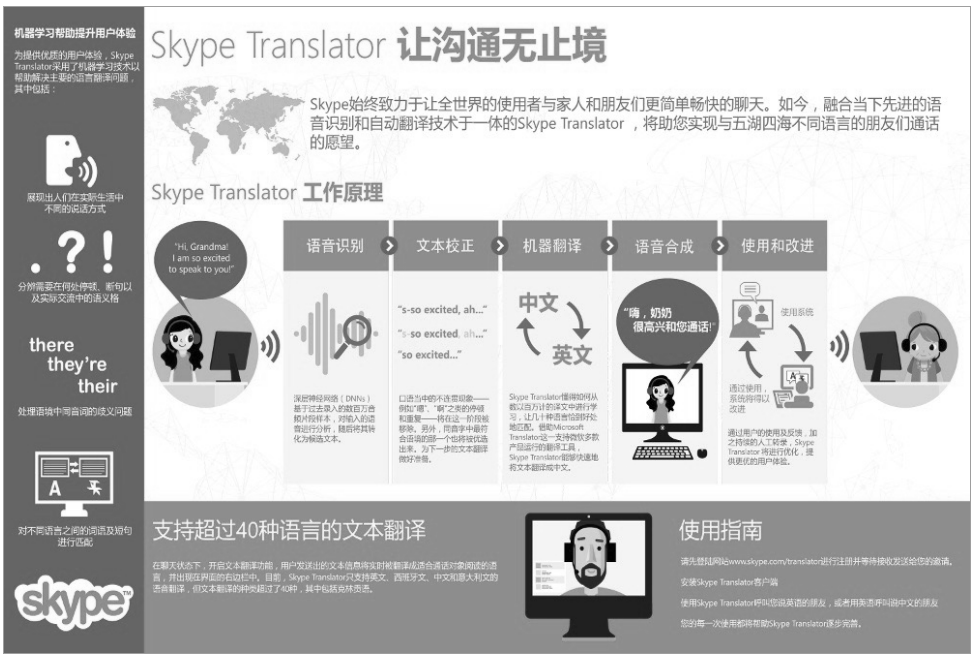

图 8.7

在准备好的数据被录入机器学习系统后，机器学习软件会在这些对话和环境涉及的单词中搭建一个统计模型。当用户说话时，软件会在该统计模型中寻找相似的内容，然后应用到预先“学到”的转换程序中，将音频转换为文本，再将文本转换成另一种语言。

虽然语音识别一直是近几十年来的重要研究课题，但是该技术的发展普遍受到错误率高、麦克风敏感度差异、噪声环境等因素的阻碍。将深层神经网络（DNNs）技术引入语音识别，极大地降低了错误率、提高了可靠性，最终使这项语音翻译技术得以广泛应用。

8.5　交互式分析[①]

在机器学习和数据挖掘领域，有“数据科学家”这样一种职位。“数据科学家”在 2009 年由 Natahn Yau 首次提出，是指能采用科学方法、运用数据挖掘工具，对复杂多量的数字、符号、文字、网址、音频或视频等信息进行数字化重现与认识，并能寻找新的数据洞察的工程师或专家（不同于统计学家或分析师）。一个优秀的数据科学家需要具备的素质包括：懂数据采集、懂数学算法、懂数学软件、懂数据分析、懂预测分析、懂市场应用、懂决策分析等。

传统的典型应用（如推荐系统）的一个数据流过程，需要经历“使用 Hadoop 做 ETL→使用 Impala/Drill 等做数据探索→使用 Tableau 做报表→使用 R 语言或者 Mahout 做高级分析→最终形成一个数据产品”等过程，如图 8.8 所示。

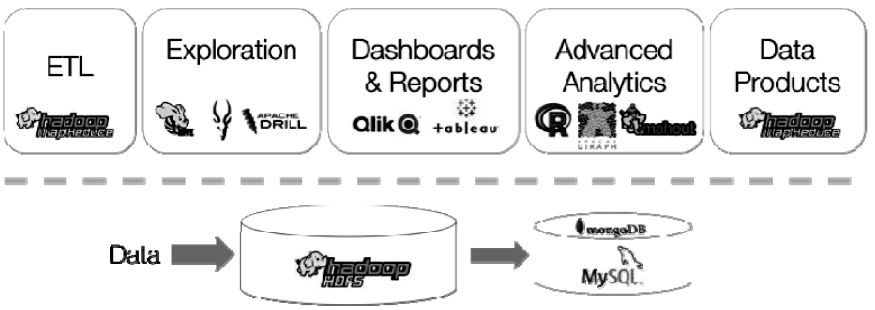

图 8.8

这个过程非常复杂，对技能要求非常高，需要懂一系列复杂的系统和工具。Databricks 创新地将 ETL、探索、高级分析、报表、数据产品统一到一个平台上，如图 8.9 所示。

这里用到的核心工具是 Notebooks。Notebooks 提供一个交互式的工作区，数据科学家可以使用 R、Python、Scala、SQL 等语言直接在工作区中进行输入，结果将以图形化的方式展现。移动设备的地理分布如图 8.10 所示。

① 参见 https://databricks.com/blog/2014/07/14/databricks-cloud-making-big-data-easy.html。

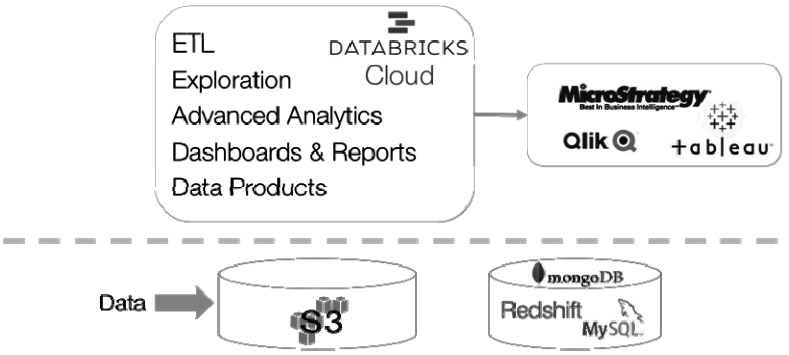

图 8.9

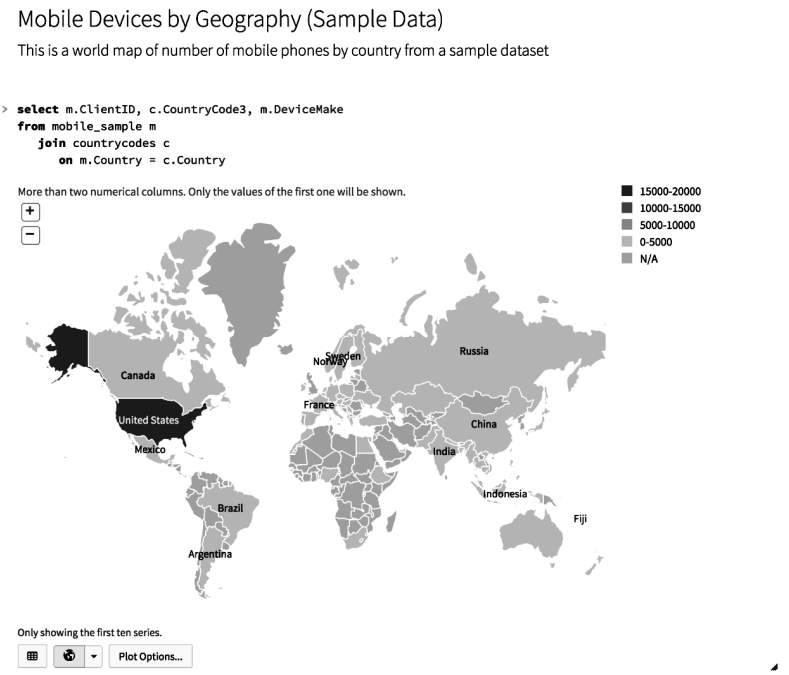

图 8.10

8.6 深度学习[①②]

8.6.1 深度学习概述

Artificial Intelligence（人工智能）是人类美好的愿望之一。虽然计算机技术已经取得了长足的进步，但截至目前，还没有一台计算机能够产生“自我”的意识。的确，在人类和大量现有数据的帮

① Deep Learning（深度学习）学习笔记整理系列，http://blog.csdn.net/zouxy09/article/details/8775360。

② 一文读懂机器学习，http://www.36dsj.com/archives/20382。

助下，计算机可以表现得十分强大，但是离开了这两者，它甚至都不能分辨两只小动物。

1950 年提出图灵试验（图灵是计算机和人工智能的鼻祖，分别对应于著名的“图灵机”和“图灵测试”）的设想，即隔墙对话，你将不知道与你对话的是人还是机器。这无疑给计算机，尤其是人工智能，预设了一个很高的期望值。但是半个世纪过去了，人工智能的进展远远没有达到图灵试验的标准。这不仅让多年翘首以待的人们心灰意冷，更让人们认为相关领域是“伪科学”。

自 2006 年以来，机器学习领域取得了突破性的进展，图灵试验至少不再那么可望而不可即。至于技术手段，不仅依赖于云计算对大数据的并行处理能力，而且依赖于算法。这个算法就是 Deep Learning。借助 Deep Learning 算法，人类终于找到了如何处理“抽象概念”这个亘古难题的方法。

2012 年 6 月，《纽约时报》披露了 Google Brain 项目，吸引了公众的广泛关注。这个项目是由斯坦福大学的机器学习教授 Andrew Ng 和大规模计算机系统方面的世界顶尖专家 Jeff Dean 共同主导的，他们用 16 000 个 CPU Core 的并行计算平台训练一种称为“深度神经网络”（Deep Neural Networks，DNN）的机器学习模型（内部共有 10 亿个节点），在语音识别和图像识别等领域获得了巨大的成功。

项目负责人之一 Andrew 称：“我们没有像通常做的那样自己框定边界，而是直接把海量数据投放到算法中，让数据自己说话，系统会自动从数据中学习。”另外一名负责人 Jeff 则说：“我们在训练的时候从来不会告诉机器‘这是一只猫’，系统其实是自己发明或者领悟了‘猫’的概念。”

2012 年 11 月，微软在中国天津的一次活动上公开演示了一个全自动的同声传译系统，讲演者用英文演讲，后台的计算机自动完成语音识别、中英机器翻译和中文语音合成，效果非常流畅。据报道，后面支撑的关键技术也是 DNN，或者深度学习（Deep Learning，DL）。

2013 年 1 月，在百度年会上，创始人兼 CEO 李彦宏高调宣布要成立百度研究院，其中第一个成立的就是“深度学习研究所”（Institute of Deep Learning，IDL）。

为什么拥有大数据的互联网公司争相投入大量资源研发深度学习技术。什么是 Deep Learning？为什么有 Deep Learning？它是怎么来的？又能做什么？目前存在哪些困难？这些问题需要慢慢解答。我们首先来了解一下机器学习的背景。

8.6.2　机器学习的背景

机器学习（Machine Learning）是一门专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构，使之不断改善自身性能的学科。机器能否像人类一样具有学习能力呢？1959 年，美国的塞缪尔（Samuel）设计了一个下棋程序，这个程序具有学习能力，它可以在不断地对弈中改善自己的棋艺。这个程序向人们展示了机器学习的能力，提出了很多令人深思的社会问题与哲学问题。

机器学习虽然发展了几十年，但仍然存在很多没有得到良好解决的问题，如图像识别、语音识别、自然语言理解、天气预测、基因表达、内容推荐等。目前通过机器学习去解决这些问题的思路如图 8.11 所示（以视觉感知为例）。

图 8.11

首先通过传感器（如 CMOS）获取数据；然后经过预处理、特征提取、特征选择；再到推理、预测或者识别；最后，也就是机器学习的部分，绝大部分工作是在这里做的，也存在很多论文和研究。

而中间的几部分（特征提取、特征选择，再到推理、预测或者识别）概括起来就是特征表达。良好的特征表达对最终算法的准确性起了非常关键的作用，而且系统主要的计算和测试工作都耗费在这一部分。但是这一部分实际上是由人工完成的，即依靠人工提取特征。

截至目前，出现了很多好的特征（好的特征应具有不变性和可区分性），如 SIFT 的出现是局部图像特征描述子研究领域一项里程碑式的工作。由于 SIFT 对尺度、旋转及一定视角和光照变化等图像变化都具有不变性，并且具有很强的可区分性，因此让很多问题的解决变为可能，但它也不是万能的。

然而，手工选取特征是一种非常费力、启发式（需要专业知识）的方法，能不能选取好很大程度上要靠经验和运气，而且它的调节需要大量的时间。既然手工选取特征性能不佳，那么，能不能自动学习一些特征呢？答案是肯定的。Deep Learning 就是设计用来做这件事情的。

那它是怎么学习的呢？怎么知道哪些特征好、哪些不好呢？我们说机器学习是一门专门研究计算机怎样模拟或实现人类的学习行为的学科，那么人类的视觉系统是怎样工作的呢？我们能不能参考人脑、模拟人脑呢？

近几十年来，认知神经科学、生物学等学科的发展，让我们对神秘而又神奇的大脑不再那么陌生，也为人工智能的发展推波助澜。

8.6.3 人脑视觉机理

1981 年的诺贝尔医学奖颁给了 David Hubel（出生于加拿大的美国神经生物学家）和 Torsten Wiesel，以及 Roger Sperry。前两位的主要贡献是“发现了视觉系统的信息处理”：可视皮层是分级的，如图 8.12 所示。

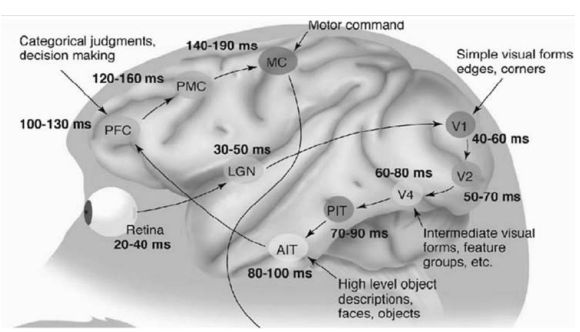

图 8.12

我们来看看他们都做了什么。1958 年，David Hubel 和 Torsten Wiesel 在 John Hopkins University 研究瞳孔区域与大脑皮层神经元的对应关系。他们首先在猫的后脑头骨上开了一个深 3 毫米的小洞，向洞里插入电极，测试神经元的活跃程度。

然后，他们在猫的眼前展现各种形状、各种亮度的物体，并且在展现每一件物体时，同时改变物体放置的位置和角度。他们期望通过这种方法，让猫的瞳孔感受不同类型、不同强弱的刺激。

之所以做这个试验，目的是去证明一个猜测：位于后脑皮层的不同视觉神经元与瞳孔所受刺激之间存在某种对应关系，一旦瞳孔受到某种刺激，后脑皮层的某一部分神经元就会活跃。经过反复试验，David Hubel 和 Torsten Wiesel 发现了一种被称为“方向选择性细胞（Orientation Selective Cell）”的神经元细胞。当瞳孔发现了眼前的物体的边缘，而且这个边缘指向某个方向时，这种神经元细胞就会活跃。

这个发现激发了人们对于神经系统的进一步思考。神经—中枢—大脑的工作过程，或许就是一个不断迭代、不断抽象的过程。

这里有两个关键词：一个是抽象，另一个是迭代。从原始信号做低级抽象，逐渐向高级抽象迭代。人类的逻辑思维经常使用高度抽象的概念。

例如，从原始信号摄入开始（瞳孔摄入像素），接着做初步处理（大脑皮层某些细胞发现边缘和方向），然后抽象（大脑判定眼前物体的形状是圆形的），最后进一步抽象（大脑进一步判定该物体是一只气球），如图 8.13 所示。

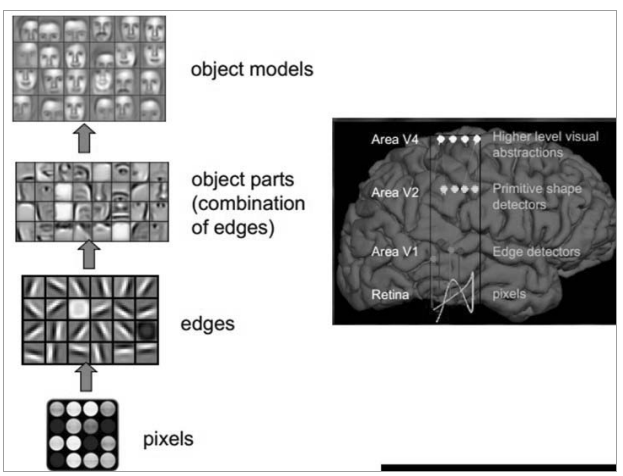

图 8.13

总的来说，人类视觉系统的信息处理是分级的。从低级的 V1 区提取边缘特征，再到 V2 区的形状或者目标等，最后到更高层，即整个目标、目标的行为等。也就是说高层特征是低层特征的组合，从低层到高层的特征表示越来越抽象，越来越能表现语义或者意图。而抽象层面越高，存在的可能猜测就越少，就越有利于分类。例如，单词集合和句子的对应关系是多对一的，句子和语义的对应关系又是多对一的，语义和意图的对应关系还是多对一的，这是一个层级体系。

敏感的人注意到一个关键词：分层。而 Deep Learning 的 Deep 是不是就表示存在很多层呢？答案是肯定的。那么，Deep Learning 是如何借鉴这个过程的呢？

因为我们要学习的是特征的表达，那么关于特征，或者说关于这个层级特征，我们需要了解得更深入一些。

8.6.4 关于特征

特征是机器学习系统的原材料，对最终模型的影响是毋庸置疑的。如果数据被很好地表达成了特征，那么线性模型通常就能达到满意的精度。对于特征，我们需要考虑哪些方面呢？

1．特征表示的粒度

学习算法在一个什么粒度上的特征表示才能发挥作用？就一张图片来说，像素级的特征根本没有价值。例如图 8.14 所示的摩托车，从像素级别根本得不到任何信息，无法进行摩托车和非摩托车的区分。而如果特征具有结构性（或者说有含义）的时候，如是否具有车把手（Handle）、是否具有车轮（Wheel），就很容易进行区分，学习算法才能发挥作用。

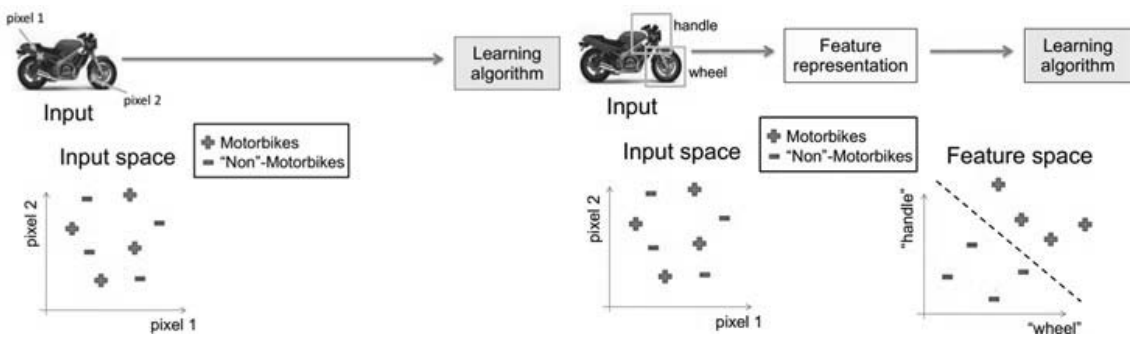

图 8.14

2．初级（浅层）特征表示

既然像素级的特征表示方法没有作用，那怎样的表示方法才有用呢？

1995 年前后，Bruno Olshausen 和 David Field 两位学者试图同时用生理学和计算机的手段研究视觉问题。他们收集了很多黑白风景照片，从这些照片中提取出 400 个小碎片，每个照片碎片的尺寸均为 16 像素×16 像素。不妨把这 400 个碎片标记为 $S[i]$，$i=0,1,\cdots,399$。接下来，从这些黑白风景照片中随机提取另一个碎片，尺寸也是 16 像素×16 像素，不妨把这个碎片标记为 T。

他们提出的问题是：如何从这 400 个碎片中选取一组碎片 $S[k]$，通过叠加的方法合成一个新的碎片，而这个新的碎片应当与随机选择的目标碎片 T 尽可能相似，同时 $S[k]$的数量尽可能少？用数学的语言来描述就是：Sum_k ($a[k]\times S[k]$) --> T，其中，$a[k]$是在叠加碎片 $S[k]$时的权重系数。

为了解决这个问题，Bruno Olshausen 和 David Field 发明了一种算法——稀疏编码（Sparse Coding）。

稀疏编码是一个重复迭代的过程，每次迭代分为两步：

（1）选择一组 $S[k]$，然后调整 $a[k]$，使得 Sum_k($a[k]$×$S[k]$) 最接近 T。

（2）固定 $a[k]$，在 400 个碎片中，选择其他更合适的碎片 $S'[k]$，替代原先的 $S[k]$，使得 Sum_k ($a[k] \times S'[k]$)最接近 T。

经过几次迭代后，最佳的 $S[k]$组合被遴选出来。令人惊奇的是，被选中的 $S[k]$基本上都是照片上不同物体的边缘线，这些线段形状相似，区别在于方向。

Bruno Olshausen 和 David Field 的算法结果与 David Hubel 和 Torsten Wiesel 的生理发现不谋而合。

也就是说，复杂图形往往由一些基本结构组成。如图 8.15 所示，一张图可以通过用 64 种正交的 edges（可以理解为正交的基本结构）来线性表示。比如，样例 x 可以用 1～64 个 edges 中的 3 个按照 0.8、0.3、0.5 的权重调和而成，而其他 edges 没有贡献，因此均为 0。

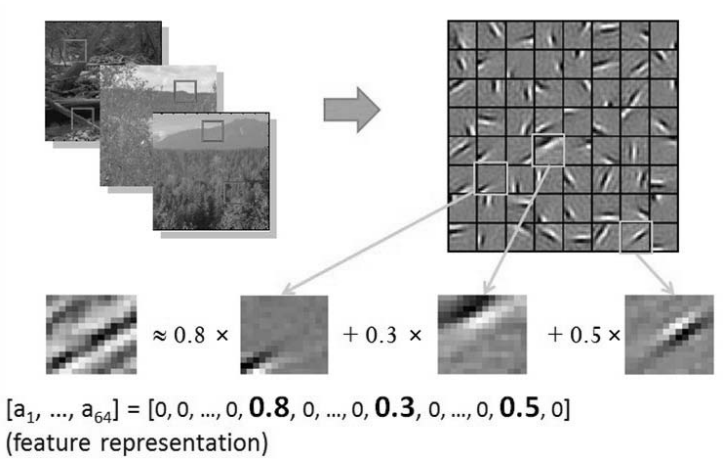

图 8.15

另外，学者还发现，不仅图像存在这个规律，声音也存在。他们从未标注的声音中发现了 20 种基本的声音结构，其余的声音都可以由这 20 种基本结构合成。

3．结构性特征表示

小块的图形可以由基本 edges 构成，那么更结构化、更复杂、具有概念性的图形如何表示呢？这就需要更高层次的特征表示，如 V2、V4。因此，V1 看像素级是像素级，V2 看 V1 是像素级，逐层递进，高层表达由低层表达组合而成。专业说法就是基（basis）。V1 层取提出的 basis 是边缘，V2 层是 V1 层这些 basis 的组合，这时候 V2 层得到的又是高一层的 basis。

8.6.5　需要有多少个特征

任何一种方法，特征越多，给出的参考信息就越多，准确性就会得到提升。但特征多意味着计算复杂、探索空间大，可以用来训练的数据在每个特征上就会稀疏，因此会带来各种问题，所以说并不是特征越多越好。

上述讨论得出一个结论：Deep Learning 需要用多层来获得更抽象的特征表达。那么多少层才合适呢？用什么架构来建模呢？怎么进行非监督训练呢？

8.6.6 深度学习的基本思想

假设有一个系统 S，它有 n 层（$S_1, S_2, \cdots, S_n$），它的输入是 I，输出是 O，形象地表示为：$I \Rightarrow S_1 \Rightarrow S2 \Rightarrow \ldots.. \Rightarrow S_n \Rightarrow O$，如果输出 O 等于输入 I，则输入 I 经过这个系统变化后没有任何信息损失。设处理 a 信息得到 b，再处理 b 得到 c，那么可以证明：a 和 c 的互信息不会超过 a 和 b 的互信息。这表明信息处理不会增加信息，大部分处理会丢失信息。现在回到主题 Deep Learning，我们需要自动地学习特征。假设有一堆输入 I（一堆图像或者文本），设计了一个系统 S（有 n 层），通过调整系统中的参数，使得它的输出仍然是输入 I，那么我们就可以自动获取得到输入 I 的一系列层次特征，即 $S_1, S_2, \cdots, S_n$。

对于深度学习来说，其基本思想就是堆叠多个层，将这一层的输出作为下一层的输入。通过这种方式就可以实现对输入信息进行分级表达。

另外，前面的假设是输出严格地等于输入。这个限制太严格，我们可以略微地放松这个限制，如使得输入与输出的差别尽可能小。这种放松会导致另外一种不同的 Deep Learning 方法。

8.6.7 浅层学习和深度学习

1．浅层学习是机器学习的第一次浪潮

20 世纪 80 年代末期，用于人工神经网络的反向传播算法（也叫 Back Propagation 算法或者 BP 算法）的发明给机器学习带来了希望，掀起了基于统计模型的机器学习热潮。人们发现，利用 BP 算法可以让一个人工神经网络模型从大量训练样本中学习统计规律，从而对未知事件做出预测。这种基于统计的机器学习方法比起过去基于人工规则的系统，在很多方面显现出优越性。这个时候的人工神经网络虽然也被称作多层感知机（Multi-Layer Perceptron），但实际上是一种只含有一层隐层节点的浅层模型。

20 世纪 90 年代，各种各样的浅层机器学习模型相继被提出，如支持向量机（SVM）、Boosting、最大熵方法（如 LR）等。这些模型的结构基本上可以看作带有一层隐层节点（如 SVM、Boosting），或没有隐层节点（如 LR）。这些模型无论是在理论分析还是在实际应用中都获得了极大的成功。相比之下，由于理论分析的难度大，训练方法又需要很多经验和技巧，这个时期的浅层人工神经网络反而相对沉寂。

2．深度学习是机器学习的第二次浪潮

2006 年，加拿大多伦多大学教授、机器学习领域的泰斗 Geoffrey Hinton 和他的学生 Ruslan Salakhutdinov 在《科学》杂志上发表了一篇文章，掀起了深度学习在学术界和工业界的浪潮。这篇文章有两个主要观点：（1）多隐层的人工神经网络具有优异的特征学习能力，学习得到的特征对数

据有更本质的刻画，从而有利于可视化或分类；（2）深度神经网络在训练上的难度可以通过“逐层初始化”来有效克服（在这篇文章中，逐层初始化是通过非监督式学习实现的）。

当前多数分类、回归等学习方法为浅层结构算法，其局限性在于，在有限样本和计算单元的情况下对复杂函数的表示能力有限，针对复杂分类问题其泛化能力受到一定制约。深度学习可以通过学习一种深层非线性网络结构，实现复杂函数逼近，表征输入数据分布式表示，并展现了强大的从少数样本集中学习数据集本质特征的能力，如图 8.16 所示。

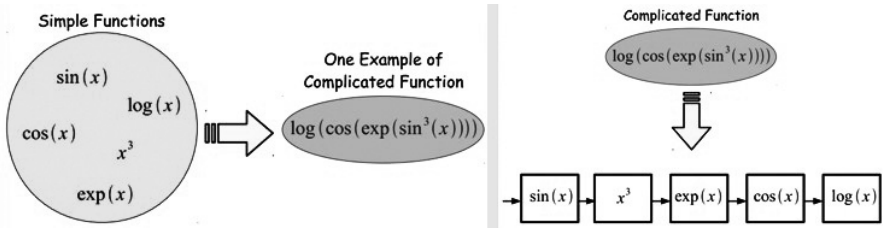

图 8.16

深度学习的实质是通过构建具有很多隐层的机器学习模型和海量的训练数据来学习更有用的特征，从而提升分类或预测的准确性。因此，“深度模型”是手段，“特征学习”是目的。区别于传统的浅层学习，深度学习的不同之处在于：（1）强调了模型结构的深度，通常有 5 层、6 层，甚至十几层的隐层节点；（2）明确突出了特征学习的重要性，也就是说，通过逐层特征变换，将样本在原空间的特征表示变换到一个新特征空间，从而使得分类或预测更加容易。与人工规则构造特征的方法相比，利用大数据来学习特征，更能够刻画出数据丰富的内在信息。

8.6.8　深度学习与神经网络

深度学习是机器学习研究中一个新的领域，其目的在于建立、模拟人脑进行分析学习的神经网络，它模仿人脑的机制来解释数据，如图像、声音和文本等。深度学习是非监督式学习的一种。

深度学习的概念来源于人工神经网络的研究。含多个隐层的多层感知器就是一种深度学习结构。深度学习通过组合低层特征形成更加抽象的高层特征来表示属性类别或特征，以发现数据的分布式特征表示。

深度学习本身是机器学习的一个分支，可以简单地理解为神经网络的发展。

二者的相同之处在于，深度学习采用了与神经网络相似的分层结构，系统由输入层、隐层（多层）、输出层组成，只有相邻的层节点之间有连接，同一层及跨层节点之间无连接，每一层都可以看作一个 Logistic Regression 模型，如图 8.17 所示。这种分层结构比较接近人脑的结构。

为了克服神经网络训练中的问题，深度学习采用了与神经网络不同的训练机制。在传统的神经网络中，采用迭代的算法来训练整个网络，随机设定初值，计算当前网络的输出，然后根据当前输出和 Label 之间的差去改变前面各层的参数，直到收敛（整体是一个梯度下降法）。而深度学习整体上是一个 layer-wise（逐层）的训练机制。这样做的原因是，如果采用 Back Propagation 的机制，那

么对于一个 Deep Network（7 层以上）来说，残差传播到最前面的层已经变得太小，就会出现所谓的 Gradient Diffusion（梯度扩散）。

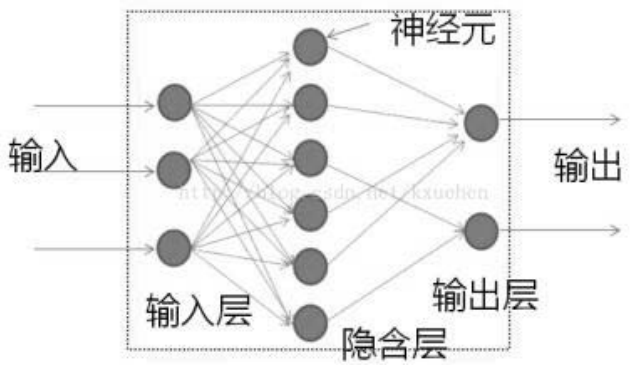

图 8.17

8.6.9 深度学习的训练过程

深度学习的训练过程具体如下：

（1）自下而上的非监督式学习（从底层开始，一层一层地往顶层训练）。

采用无标定数据（或有标定数据）分层训练各层参数。这一步可以看作一个无监督训练过程，是和传统神经网络区别最大的部分（这个过程可以看作 Feature Learning 过程）。

具体而言，先用无标定数据训练第一层，训练时先学习第一层的参数（这一层可以看作得到一个使得输出和输入差别最小的三层神经网络的隐层），由于模型容量限制及稀疏性约束，使得得到的模型能够学习到数据本身的结构，从而得到比输入更具有表示能力的特征；在学习得到第 $n-1$ 层后，将第 $n-1$ 层的输出作为第 n 层的输入，训练第 n 层，由此分别得到各层的参数。

（2）自顶向下的监督式学习（通过带标签的数据去训练，误差自顶向下传输，对网络进行微调）。

基于第一步得到的各层参数进一步微调整个多层模型的参数，这一步是一个有监督训练过程。第一步类似神经网络的随机初始化初值过程，由于深度学习的第一步不是随机初始化，而是通过学习输入数据的结构得到的，因而这个初值更接近全局最优，从而能够取得更好的效果。所以深度学习效果好很大程度上归功于第一步的 Feature Learning 过程。

8.6.10 深度学习的框架

深度学习还处于一个蓬勃发展的阶段，各种深度学习框架层出不穷。常见的深度学习框架有 Tensorflow、Caffe、Theano、Torch、Deeplearning4j、Marvin、ConvNetJS 和 MXNet 等。

下面来看一下 Caffe 架构，以便对深度学习框架有一个基本的认识。

1．Caffe[①]背景

Caffe 是一个清晰而高效的深度学习框架，它是纯粹的 C++/CUDA 架构，支持命令行、Python 和 MATLAB 接口，可以在 CPU 和 GPU 之间无缝切换。

Caffe 中用到的主要的库有如下几个。

- Google Logging Library (Glog)：一个 C++语言的应用级日志记录框架。
- LebelDB（数据存储）：是一个 Google 实现的非常高效的 KV 数据库，单进程操作。
- CBLAS Library：CPU 版本的矩阵操作。
- CUBLAS Library：GPU 版本的矩阵操作。

2．Caffe 架构[②]

Caffe 是非常模块化的，这与神经网络本身就比较模块化相关。下面简要介绍几个常用的模块。

1）Blob

Blob 是一个四维连续数组，如果使用(n,c,h,w)表示的话，那么每一维的意思分别如下。

- n（number）：输入数据量。
- c（channel）：如果是图像数据，则可以认为是通道数量。
- h,w（height,width）：如果是图像数据，则可以认为是图片的高度和宽度。

当然，Blob 不一定就是用来表示图像输入数据的。

Blob 内部有两个字段：data 和 diff。data 表示流动数据（输出数据），而 diff 则存储 BP 的梯度。data/diff 可以存储于 CPU，也可以存储于 GPU。如果某个 Layer 不支持 GPU，那么就需要将 GPU 数据复制到 CPU 上，从而造成性能开销。对于 Python/NumPy 用户来说，可以用 reshape()函数来转换为 Blob：data = data.reshape((-1,c,h,w))。

2）Layer

Caffe 提供了许多内置 Layer，每个 Layer 的输入称为 bottom blob，输出称为 top blob，如图 8.18 所示。

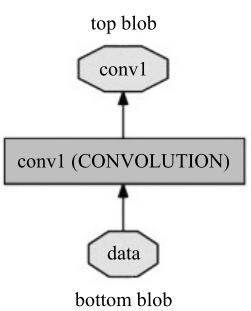

图 8.18

① Caffe 框架，http://caffe.berkeleyvision.org/。

② 参考 http://dirlt.com/caffe.html。

每层定义三个主要的计算过程：初始化、前向传播、反向传播/计算梯度。

3）Net

Net 是由 Layers 组成的 DAG，并且可以使用文本格式来描述。下面文本生成的是 Logistic Regression。

```
name: "LogReg"
layers {
  name: "mnist"
  type: DATA
  top: "data"
  top: "label"
  data_param {
    source: "input_leveldb"
    batch_size: 64
  }
}
layers {
  name: "ip"
  type: INNER_PRODUCT
  bottom: "data"
  top: "ip"
  inner_product_param {
    num_output: 2
  }
}
layers {
  name: "loss"
  type: SOFTMAX_LOSS
  bottom: "ip"
  bottom: "label"
  top: "loss"
}
```

Net 有一个初始化函数 Init()，它有两个作用：（1）创建 Blobs 和 Layers；（2）调用 Layers 的 SetUp 函数来初始化 Layers。Net 还有两个函数 Forward 和 Backward，分别调用各个 Layers 的 Forward 和 Backward。如果需要进行预测，则需先填充好 Input Blobs，然后调用 Forward 函数，最后获取 Output Blobs 作为预测结果。代码如下：

```
I0902 22:52:17.931977 2079114000 net.cpp:39] Initializing net from parameters:
name: "LogReg"
[...model prototxt printout...]
# construct the network layer-by-layer
I0902 22:52:17.932152 2079114000 net.cpp:67] Creating Layer mnist
I0902 22:52:17.932165 2079114000 net.cpp:356] mnist -> data
I0902 22:52:17.932188 2079114000 net.cpp:356] mnist -> label
I0902 22:52:17.932200 2079114000 net.cpp:96] Setting up mnist
I0902 22:52:17.935807 2079114000 data_layer.cpp:135] Opening leveldb input_leveldb
I0902 22:52:17.937155 2079114000 data_layer.cpp:195] output data size: 64,1,28,28
I0902 22:52:17.938570 2079114000 net.cpp:103] Top shape: 64 1 28 28 (50176)
```

```
I0902 22:52:17.938593 2079114000 net.cpp:103] Top shape: 64 1 1 1 (64)
I0902 22:52:17.938611 2079114000 net.cpp:67] Creating Layer ip
I0902 22:52:17.938617 2079114000 net.cpp:394] ip <- data
I0902 22:52:17.939177 2079114000 net.cpp:356] ip -> ip
I0902 22:52:17.939196 2079114000 net.cpp:96] Setting up ip
I0902 22:52:17.940289 2079114000 net.cpp:103] Top shape: 64 2 1 1 (128)
I0902 22:52:17.941270 2079114000 net.cpp:67] Creating Layer loss
I0902 22:52:17.941305 2079114000 net.cpp:394] loss <- ip
I0902 22:52:17.941314 2079114000 net.cpp:394] loss <- label
I0902 22:52:17.941323 2079114000 net.cpp:356] loss -> loss
# set up the loss and configure the backward pass
I0902 22:52:17.941328 2079114000 net.cpp:96] Setting up loss
I0902 22:52:17.941328 2079114000 net.cpp:103] Top shape: 1 1 1 1 (1)
I0902 22:52:17.941329 2079114000 net.cpp:109]     with loss weight 1
I0902 22:52:17.941779 2079114000 net.cpp:170] loss needs backward computation.
I0902 22:52:17.941787 2079114000 net.cpp:170] ip needs backward computation.
I0902 22:52:17.941794 2079114000 net.cpp:172] mnist does not need backward
computation.
# determine outputs
I0902 22:52:17.941800 2079114000 net.cpp:208] This network produces output loss
# finish initialization and report memory usage
I0902 22:52:17.941810 2079114000 net.cpp:467] Collecting Learning Rate and Weight
Decay.
I0902 22:52:17.941818 2079114000 net.cpp:219] Network initialization done.
I0902 22:52:17.941824 2079114000 net.cpp:220] Memory required for data: 201476
```

如果阅读 Caffe/Models 就会发现，这些例子下面有 train.prototxt 和 deploy.prototxt，差别仅仅在于 deploy.txt 没有 data-layer[①]，而是指定输入的 shape[②]。

```
input: "data"
input_dim: 10
input_dim: 1
input_dim: 28
input_dim: 28
```

从字面上来看，train.prototxt 是用来训练出模型的，而 deploy.prototxt 则是用来进行预测的。如下是使用 Python 进行预测的代码：

```
caffe.set_mode_cpu()
net = caffe.Net('caffe-conf/test.prototxt',
             'uv_iter_10000.caffemodel',
             caffe.TEST)
data = data.reshape((-1,1,28,28))
out = net.forward_all(**{'data': data})
rs = out['prob'] # 得到的是 softmax
print_timer("predict")
```

① data_layer：网络的底层，主要用于将数据经 Blob 传入网络中。

② Shape：存放形状的数据结构。

4）Solver

下面是 solver.prototxt 的一个示例（从 examples/mnist/修改而来）：

```
# The train/test net protocol buffer definition
net: "caffe-conf/train.prototxt"
# 如果 test 数据量是 10000，而 bacth_size = 100，那么 test_iter 就应该设置为 100
# 这样每次进行 test 就可以使用所有的 cases
test_iter: 90
# Carry out testing every 500 training iterations.
# 每进行 500 轮迭代进行一次测试
test_interval: 500
# 下面这些是训练所用的参数
# The base learning rate,momentum and the weight decay of the network.
base_lr: 0.01
momentum: 0.9
weight_decay: 0.0005
# The learning rate policy
lr_policy: "inv"
gamma: 0.0001
power: 0.75
# Display every 100 iterations
display: 500
# The maximum number of iterations
max_iter: 10000
# snapshot intermediate results
# 每进行 500 轮迭代进行一次 snapshot
# 每一轮使用的数据量大小为 batch_size
snapshot: 500
snapshot_prefix: "uv"
snapshot_after_train: true
# solver mode: CPU or GPU
# 使用 CPU 训练
solver_mode: CPU
```

“net”表示 train 和 test 使用同一个 Net。在 net.prototxt 中可以使用 include 语法来声明某个 Layer 是否需要包含在 train/test 阶段。

如果在训练时不想进行 test，那么可以指定上面的“net”为“train_net”。当然也可以使用“test_nets”来指定多个 test_net。

5）Python 接口[①]

Caffe 支持三种接口：命令行、Python 和 MATLAB。这里只介绍 Python 接口。

① http://caffe.berkeleyvision.org/tutorial/interfaces.html

- caffe.Net：加载、配置和执行模型的主要接口。
- caffe.Classsifier 和 caffe.Detector：给普通任务提供方便的接口。
- caffe.io：负责预处理输入、输出和通信缓存。
- caffe.draw：提供网络架构的可视化。
- Caffe blobs：提供多维矩阵易用接口。

3．Caffe 使用案例：使用数据库 CIFAR-10[①]

60 000 张 32 像素×32 像素的彩色图像 10 类，50 000 张用于训练，10 000 张用于测试，如图 8.19 所示。

图 8.19

1）准备

在终端运行以下指令：

```
cd $CAFFE_ROOT/data/cifar10
./get_cifar10.sh
cd $CAFFE_ROOT/examples/cifar10
./create_cifar10.sh
```

其中，CAFFE_ROOT 是 Caffe-Master 在主机上的地址。

运行之后，将会在 examples 目录中出现数据库文件./cifar10-leveldb 和数据库图像均值二进制文件./mean.binaryproto，如图 8.20 所示。

① 案例 http://caffe.berkeleyvision.org/gathered/examples/cifar10.html。

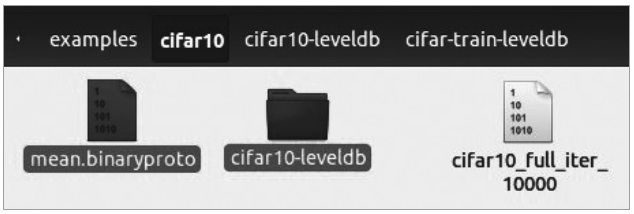

图 8.20

2）模型

该 CNN 由卷积层、POOLing 层、非线性变换层、在顶端的局部对比归一化线性分类器组成。该模型的定义在 CAFFE_ROOT/examples/cifar10_quick_train.prototxt 文件中，可以进行修改。

3）训练和测试

训练这个模型非常简单，在写好参数设置的文件 cifar10_quick_solver.prototxt 及定义的文件 cifar10_quick_train.prototxt 和 cifar10_quick_test.prototxt 以后，运行 train_quick.sh，或者在终端输入如下命令即可：

```
cd $CAFFE_ROOT/examples/cifar10
./train_quick.sh
```

train_quick.sh 是一个简单的脚本，它会把执行的信息显示出来。训练的工具是 train_net.bin，以 cifar10_quick_solver.prototxt 作为参数。然后出现类似如下信息：

```
I0317 21:52:48.945710 2008298256 net.cpp:74] Creating Layer conv1
I0317 21:52:48.945716 2008298256 net.cpp:84] conv1 <- data
I0317 21:52:48.945725 2008298256 net.cpp:110] conv1 -> conv1
I0317 21:52:49.298691 2008298256 net.cpp:125] Top shape: 100 32 32 32 (3276800)
I0317 21:52:49.298719 2008298256 net.cpp:151] conv1 needs backward computation.
```

接着：

```
0317 21:52:49.309370 2008298256 net.cpp:166] Network initialization done.
I0317 21:52:49.309376 2008298256 net.cpp:167] Memory required for Data 23790808
I0317 21:52:49.309422 2008298256 solver.cpp:36] Solver scaffolding done.
I0317 21:52:49.309447 2008298256 solver.cpp:47] Solving CIFAR10_quick_train
```

之后，训练开始：

```
I0317 21:53:12.179772 2008298256 solver.cpp:208] Iteration 100,lr = 0.001
I0317 21:53:12.185698 2008298256 solver.cpp:65] Iteration 100,loss = 1.73643
...
I0317 21:54:41.150030 2008298256 solver.cpp:87] Iteration 500,Testing net
I0317 21:54:47.129461 2008298256 solver.cpp:114] Test score #0: 0.5504
I0317 21:54:47.129500 2008298256 solver.cpp:114] Test score #1: 1.27805
```

其中，每 100 次迭代显示一次训练时 lr（learningrate）和 loss（训练损失函数），每 500 次测试一次，输出 score 0（准确率）和 score 1（测试损失函数）。当 5000 次迭代之后，准确率约为 75%，模型的参数以二进制 Protobuf 格式存储在 cifar10_quick_iter_5000 中，然后这个模型就可以用来运行在新数据上了。

4）对比 CPU/GPU

另外，更改 cifar*solver.prototxt 文件可以使用 CPU 训练，可以对比 CPU 训练和 GPU 训练的差别。

```
# solver mode: CPU or GPU
solver_mode: CPU
```

8.6.11　深度学习与 GPU

深度学习通过构建深层神经网络来模拟人脑的工作原理。如图 8.21 所示，深层神经网络由一个输入层、数个隐层及一个输出层构成。每层有若干个神经元，神经元之间有连接权重。每个神经元模拟人类的神经细胞，而节点之间的连接模拟神经细胞之间的连接。

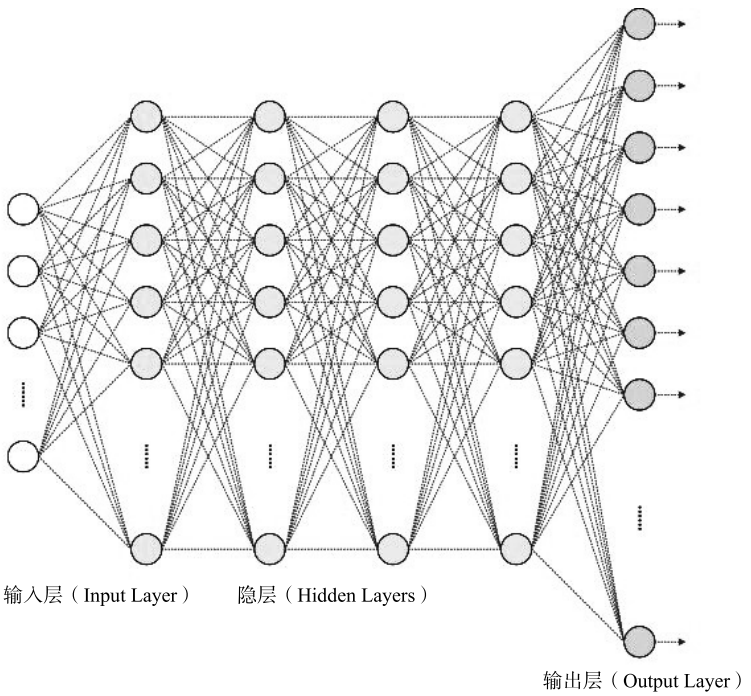

图 8.21

深层神经网络模型复杂，训练数据多，计算量大。一方面，DNN 需要模拟人脑的计算能力，而人脑包含 100 多亿个神经细胞，这要求 DNN 中的神经元非常多，神经元间的连接数量也相当惊人。从数学的角度看，DNN 中的每个神经元都包含数学计算（如 Sigmoid、ReLu 或者 Softmax 函数），需要估计的参数量也极大。在语音识别和图像识别应用中，神经元达数万个，参数达数千万个，模型复杂导致计算量大。另一方面，DNN 需要大量数据才能训练出高准确率的模型。DNN 参数量大、模型复杂，为了避免过拟合，需要海量训练数据。两方面因素叠加，导致训练一个模型耗时惊人。以语音识别为例，目前业界通常使用的样本量达数十亿，用 CPU 单机需要数年才能完成一次训练，用流行的 GPU 卡也需要数周才能完成一次训练。

图 8.22 是 CPU 和 GPU 对比基准测试，相比 CPU，借助 GPU 执行任务的速度可以提升几十倍。

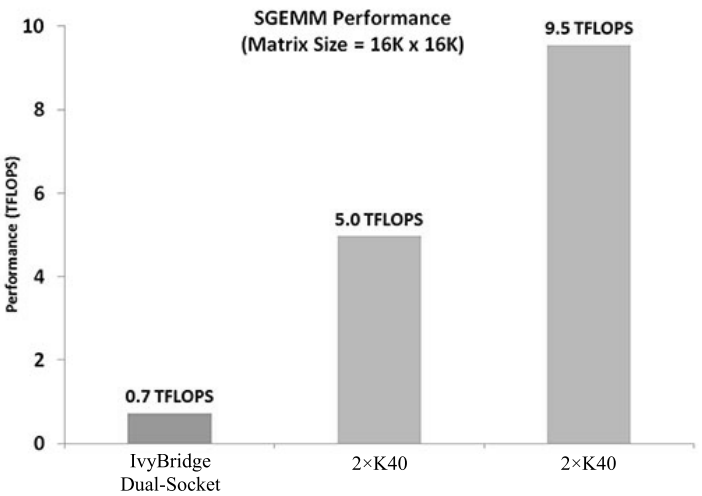

图 8.22

GPU 出色的浮点计算性能显著提高了深度学习的两大关键活动——分类和卷积的性能，同时又达到了所需的精准度。NVIDIA 表示，深度学习需要很高的内在并行度、大量的浮点计算能力及矩阵预算，而 GPU 可以提供这些能力，并且在相同的精度下，相比传统的 CPU 方式，拥有更快的处理速度、更少的服务器投入和更低的功耗。

而 GPU 和 CPU 没有本质的区别，主要区别在于 CPU 的目标是让用户有更短的响应时间，即在编辑文档或者浏览网页的时候，用最短的时间对鼠标、键盘操作做出响应。要达到这个目的，最重要的是单线程的处理能力，所以在进行芯片设计的时候，里面有大量的单元来保证单线程处理性能。CPU 有大量的资源做分级预测或者单线程寻找并行性的工作，总体来说就是 LU，即真正实现浮点整行运算的公共单元比例比 GPU 少。

另外，GPU 计算在设计的时候要保证做图形图像渲染有很好的性能。图形图像渲染任务与其他领域的计算或者通用计算的差别不是很大。以卷积为例，比如做二维的卷积，输入的是图像，竖直的角度是一个矩阵，卷积核每个元素和输入图像进行相应的乘加，放在输出的二维矩阵里。

除了 GPU 外，为了解决深度学习计算效率的问题，业界还有一种思路，即打造深度学习专用芯片。最典型的是 IBM 研发出一种新型芯片 SyNAPSE，它虽然拥有 100 万个神经元芯片、2.56 亿个突触、4096 个核心及 54 亿个晶体管，但它的功率仅有 63mW。这种新芯片可以伸缩扩展，数个芯片连接在一起就可以组成强大的神经网络。

8.6.12 深度学习小结与展望

1. 小结

深度学习算法自动提取分类所需的低层次或者高层次特征。高层次特征是指该特征可以分级（层

次）地依赖其他特征。例如，对于机器视觉，深度学习算法从原始图像去学习得到它的一个低层次表达，如边缘检测器、小波滤波器等，然后在这些低层次表达的基础上再建立表达，如这些低层次表达的线性或者非线性组合，然后重复这个过程，最后得到一个高层次的表达。

深度学习能够得到更好地表示数据的特征，同时由于模型的层次、参数很多，容量足够，因此，模型有能力表示大规模数据。所以对于图像、语音这种特征不明显（需要手工设计且很多没有直观的物理含义）的问题，能够在大规模训练数据上取得更好的效果。此外，从模式识别特征和分类器的角度来看，深度学习框架将特征和分类器结合到一个框架中，用数据去学习特征，在使用中减少了手工设计特征的巨大工作量，因此，不仅效果更好，而且使用起来也有很多方便之处。

当然，深度学习本身并不是完美的，也不是解决任何机器学习问题的利器，不应该被放大到一个无所不能的程度。

2．未来

深度学习目前的关注点还是从机器学习领域借鉴一些可以在深度学习中使用的方法，特别是降维领域。例如，目前的一项工作就是稀疏编码，通过压缩感知理论对高维数据进行降维，使用元素非常少的向量就可以精确地代表原来的高维信号。另一项工作就是半监督流行学习，通过测量训练样本的相似性，将高维数据的这种相似性投影到低维空间。此外，深度学习还有很多核心问题需要解决：

（1）对于一个特定的框架来说，对于多少维的输入，它可以表现得较优？

（2）对捕捉短时或者长时间的时间依赖，哪种架构才是最有效的？

（3）对于一个给定的深度学习架构，如何融合多种感知的信息？

（4）有什么正确的机理可以增强一个给定的深度学习架构，以改进其鲁棒性及对扭曲和数据丢失的不变性？

（5）模型方面是否有其他更为有效且有理论依据的深度模型学习算法？

探索新的特征提取模型是值得深入研究的内容。此外，有效的可并行训练算法也是值得研究的一个方向。当前基于最小批处理的随机梯度优化算法很难在多计算机中进行并行训练，通常的解决办法是利用图形处理单元加速学习过程。然而单个机器 GPU 对大规模数据识别或相似任务数据集并不适用。在深度学习的应用拓展方面，如何合理且充分地利用深度学习增强传统学习算法的性能仍是目前各领域的研究重点。

8.7　小结

本章主要介绍了机器学习，尤其是当前最热门的深度学习。深度学习可以说掀起了人工智能的又一次热潮，但是大家要清楚地认识到，这离真正的 AI（人工智能）还差得很远。但总的来说，我们离电影中描述的未来世界更近了一步，不是吗？