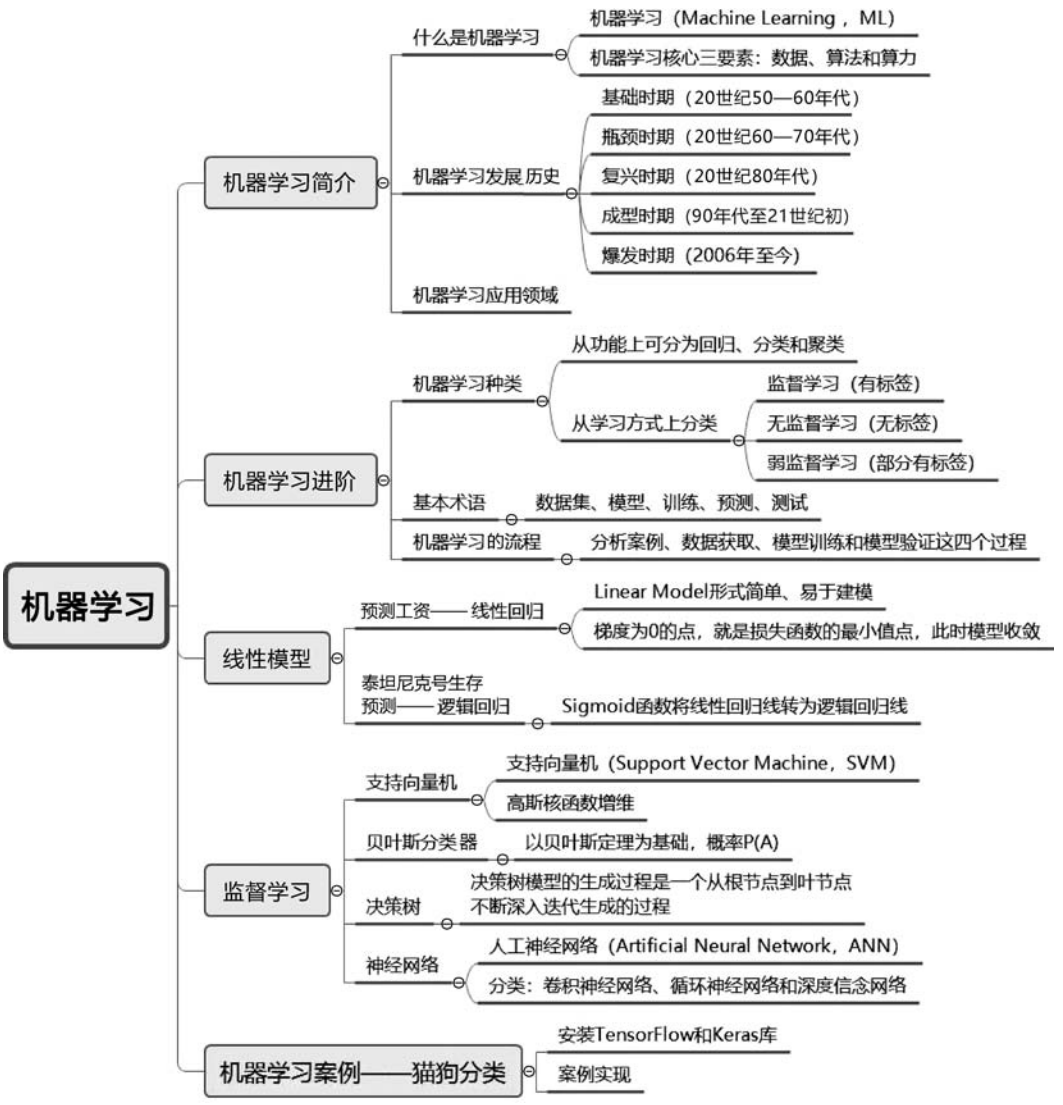
第 3 章

CHAPTER 3

机 器 学 习

本章思维导图

视频讲解



本章目标

- 了解机器学习的发展历史。
- 了解机器学习的应用领域。
- 掌握机器学习的基本术语和概念。
- 掌握机器学习的基本流程。
- 熟悉机器学习中各种模型的工作原理。

3.1　机器学习简介


视频讲解

3.1.1　什么是机器学习

机器学习(Machine Learning,ML)是人工智能的核心分支,是让机器自己做主进行学习,使机器模拟或实现人类的学习行为,以获取新的知识或技能,并能够重新组织已有的知识结构,不断改善自身的性能。在学习过程中,不需要告诉计算机具体做什么,只告诉计算机做成什么样子,让计算机“自己看着办”,学会“察言观色”,到时候给出满意的解决方案就行。机器学习的理论主要是设计和分析一些让计算机可以自动“学习”的算法,能够从数据中自动分析获得规律,并利用规律对未知数据进行预测的算法。因此,机器学习的核心就是数据、算法(模型)和算力(计算机计算能力)。

机器学习作为人工智能的一个独立方向,正处于高速发展之中。机器学习应用领域十分广泛,如图 3-1 所示,涉及数据挖掘、统计学习、计算机视觉、自然语言处理、生物特征识别、搜索引擎、医学诊断、检测信用卡欺诈、证券市场分析、DNA 序列测序、语音和手写识别、战略游戏和机器人运用等。目前,机器学习还处于弱人工智能阶段,并不会让机器产生意识。

图 3-1　机器学习范畴


视频讲解

3.1.2　机器学习发展历史

最早的机器学习算法可以追溯到 20 世纪初,至今已经过去了 100 多年。从 1980 年机器学习成为一个独立的方向开始算起,至今也已经过去了 40 年。在这 100 多年中,经过一

代又一代人的努力，诞生了大量经典的方法，具体成果如表 3-1 所示。

表 3-1 机器学习发展时间表

时 间 段	机器学习理论	代表性成果
20 世纪 50 年代初	人工智能研究处于推理期	A. Newell 和 H. Simon 的“逻辑理论家”(Logic Theorist)程序证明了数学原理，以及此后的“通用问题求解”(General Problem Solving)程序
	已出现机器学习的相关研究	1952 年，阿瑟·萨缪尔(Arthur Samuel)在 IBM 公司研制了一个西洋跳棋程序，这是人工智能下棋问题的由来
20 世纪 50 年代中后期	开始出现基于神经网络的“连接主义”(Connectionism)学习	F. Rosenblatt 提出了感知机(Perceptron)，但该感知机只能处理线性分类问题，处理不了“异或”逻辑。还有B. Widrow 提出的 Adaline
20 世纪 60 年代至 70 年代	基于逻辑表示的“符号主义”(Symbolism)学习技术蓬勃发展	P. Winston 的结构学习系统，R. S. Michalski 的基于逻辑的归纳学习系统，以及 E. B. Hunt 的概念学习系统
	以决策理论为基础的学习技术	
	强化学习技术	N. J. Nilson 的“学习机器”
	统计学习理论的一些奠基性成果	支持向量、VC 维、结构风险最小化原则
20 世纪 80 年代至 90 年代中期	机械学习(死记硬背式学习) 示教学习(从指令中学习) 类比学习(通过观察和发现学习) 归纳学习(从样例中学习)	学习方式分类
	从样例中学习的主流技术之一： (1) 符号主义学习 (2) 基于逻辑的学习	(1) 决策树(Decision Tree) (2) 归纳逻辑程序设计(Inductive Logic Programming，ILP)具有很强的知识表示能力，可以较容易地表达出复杂的数据关系，但会导致学习过程面临的假设空间太大，复杂度极高，因此，问题规模稍大就难以有效地进行学习
	从样例中学习的主流技术之二：基于神经网络的连接主义学习	1983 年，J. J. Hopfield 利用神经网络求解“流动推销员问题”这个 NP 难题。1986 年，D. E. Rumelhart 等人重新发明了 BP 算法，BP 算法一直是应用最广泛的机器学习算法之一
	20 世纪 80 年代是机器学习成为一个独立的学科领域，各种机器学习技术百花初绽的时期	连接主义学习的最大局限是“试错性”，学习过程涉及大量参数，而参数的设置缺乏理论指导，主要靠手工“调参”，参数调节失之毫厘，学习结果可能谬以千里
20 世纪 90 年代中期	统计学习(Statistical Learning)	支持向量机(Support Vector Machine，SVM)，核方法(Kernel Methods)
21 世纪初至今	深度学习(Deep Learning)	深度学习兴起的原因：数据量大，机器计算能力强

追溯历史，就会发现机器学习的技术爆发有其历史必然性，属于技术发展的必然产物。而理清机器学习的发展脉络有助于整体把握机器学习或者人工智能的技术框架，有助于从“道”的层面理解这一技术领域。机器学习的发展史如图 3-2 所示。

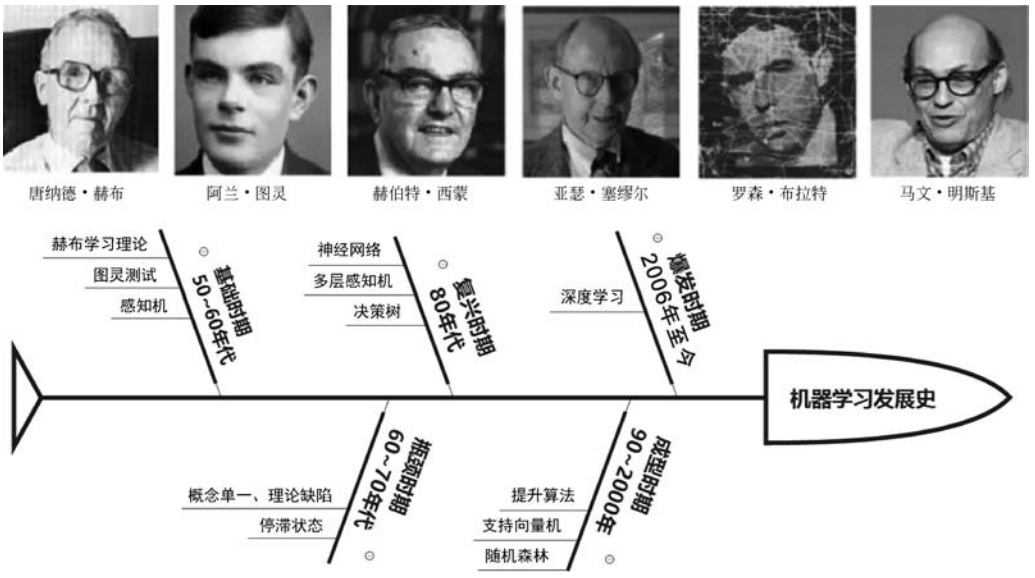


图 3-2　机器学习发展史

1. 诞生并奠定基础时期

1949 年，赫布基于神经心理学提出了一种学习方式，被称为“赫布学习理论”。该理论如此描述：假设反射活动的持续性或反复性会导致细胞的持续性变化并增加其稳定性，当一个神经元 A 能持续或反复激发神经元 B 时，这两个神经元或其中一个便会发生某些生长过程或代谢变化，致使 A 作为能使 B 兴奋的细胞之一，A 的效能增强了。赫布学习理论解释了神经元如何组成连接，从而形成记忆印痕。从整体的角度来看，赫布学习理论是神经网络（Neural Network，NN）形成记忆痕迹的首要基础，也是最简单的神经元（Neuron）学习规则。

1950 年，艾伦·麦席森·图灵提出图灵测试来判定计算机是否智能。如图 3-3 所示，图灵测试让机器与人对话，能够令人信服地说明“思考的机器”是可能的。2014 年 6 月 8 日，一个叫作尤金·古斯特曼的聊天机器人成功让人类相它是一个 13 岁的男孩，成为有史以来首台通过图灵测试的计算机。这被认为是人工智能发展的一个里程碑事件。

1952 年，IBM 科学家亚瑟·塞缪尔开发了一个跳棋程序。该程序能够通过观察当前位置，并学习一个隐含的模型，从而为后续动作提供更好的指导。塞缪尔发现，伴随着该游戏程序运行时间的增加，其可以实现越来越好的后续指导。通过这个程序，塞缪尔驳倒了普罗维登斯提出的机器无法超越人类，像人类一样写代码和学习的模式。塞缪尔创造了“机器学习”这一术语，如图 3-4 所示。

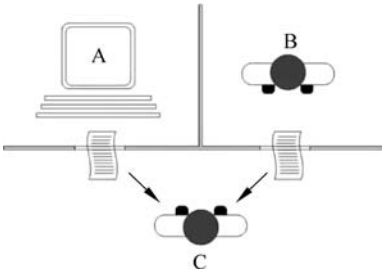


图 3-3　图灵测试

图 3-4　塞缪尔(图片来源：Brief History of Machine Learning)

1957 年，罗森·布拉特基于神经感知科学背景提出了第二模型，非常类似于今天的机器学习模型。这在当时是一个令人兴奋的发现，比赫布的想法适用性更强。基于这个模型，罗森·布拉特设计出了第一个计算机神经网络——感知机(Perceptron)，模拟了人脑的运作方式。罗森·布拉特对感知机的定义如下：感知机旨在说明一般智能系统的一些基本属性，不会被个别特例或通常不知道的东西束缚，也不会因为那些个别生物有机体的情况而陷入混乱。3 年后，维德罗首次使用 Delta 学习规则(即最小二乘法)用于感知器的训练步骤，创造了一个良好的感知机线性分类器，如图 3-5 所示。

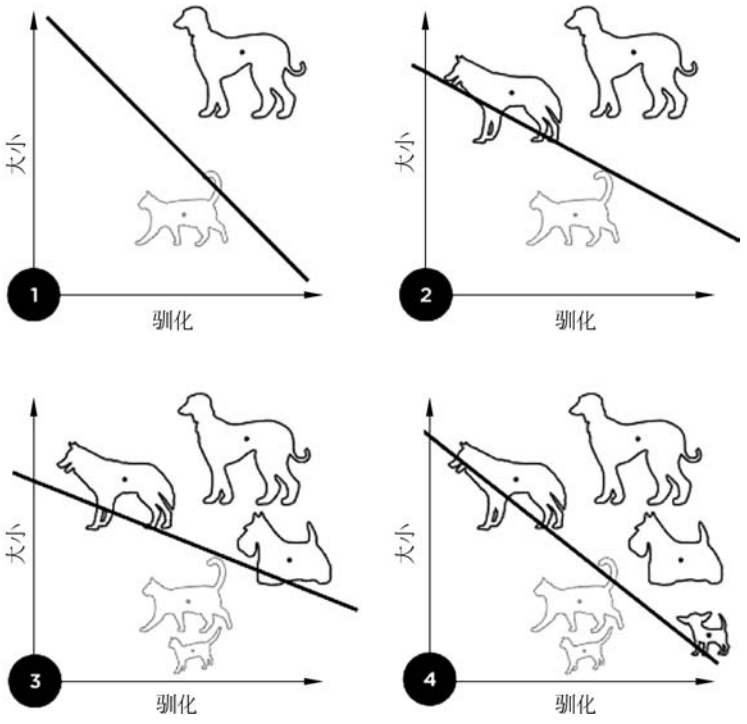


图 3-5　感知机线性分类器

1967 年，最近邻算法(the Nearest Neighbor algorithm)，或者说 K 最近邻(KNN，K-Nearest Neighbor)分类算法出现，使计算机可以进行简单的模式识别。所谓 K 最近邻，就是 K 个最近的邻居的意思，说的是每个样本都可以用它最接近的 K 个邻居来代表。KNN

算法的核心思想是：如果一个样本在特征空间中的 K 个最相邻的样本中的大多数属于某一个类别，那么该样本也属于这个类别，并具有这个类别上样本的特性。这就是所谓的“少数听从多数”原则，如图 3-6 所示。

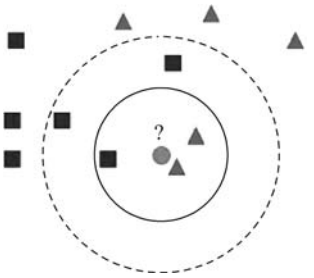


图 3-6　KNN 算法

1969 年，马文·明斯基提出了著名的异或(XOR)问题，指出感知机在线性不可分的数据分布上是失效的，如图 3-7 所示。此后神经网络的研究者进入了“寒冬”，直到 1980 年才再一次复苏。

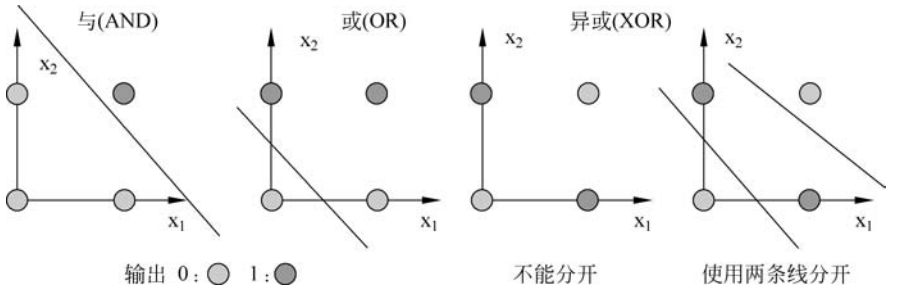


图 3-7　XOR 问题——数据线性不可分

2. 停滞不前的瓶颈时期

从 20 世纪 60 年代中期到 70 年代末，机器学习的发展步伐几乎处于停滞状态。无论是理论研究还是计算机硬件限制，使得整个人工智能领域的发展都遇到了很大的瓶颈。虽然这个时期温斯顿(Winston)的结构学习系统和海斯·罗思(Hayes Roth)的基于逻辑的归纳学习系统取得了较大的进展，但只能学习单一概念，而且未能投入实际应用。此外，神经网络学习机因理论缺陷也未能达到预期效果而转入低潮。

3. 希望之光重新点亮的复兴时期

伟博斯在 1981 年的神经网络 BP(Back Propagation，反向传播)算法中具体提出多层感知机模型，如图 3-8 所示。虽然 BP 算法早在 1970 年就已经以“自动微分的反向模型”(reverse mode of automatic differentiation)为名提出来了，但直到此时才真正发挥效用。在 1985—1986 年，神经网络研究人员鲁梅尔哈特、辛顿、威廉姆斯-赫、尼尔森继提出了使用 BP 算法训练的多参数线性规划(MLP)的理念，成为后来深度学习的基石。直到今天，BP 算法仍然是神经网络架构的关键因素。有了这些新思想，神经网络的研究又加快了。

在另一个谱系中，昆兰于 1986 年提出了一种非常出名的机器学习算法——决策树，也称为 ID3 算法。决策树是另一个主流机器学习算法的突破点，该算法也被发布成为一款软件，能以简单的规划和明确的推论找到更多的现实案例，而这一点正好和神经网络黑箱模型相反。决策树算法流程如图 3-9 所示。

决策树是一个预测模型，代表的是对象属性与对象值之间的一种映射关系。树中每个节点表示某个对象，而每个分叉路径则代表的某个可能的属性值，而每个叶节点则对应从根节点到该叶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，若要有复数输出，可以建立独立的决策树以处理不同输出。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。

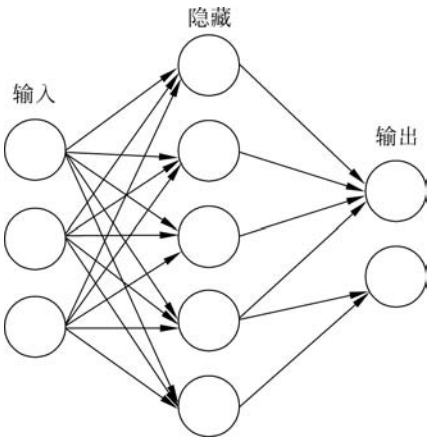


图 3-8　多层感知机(或人工神经网络)

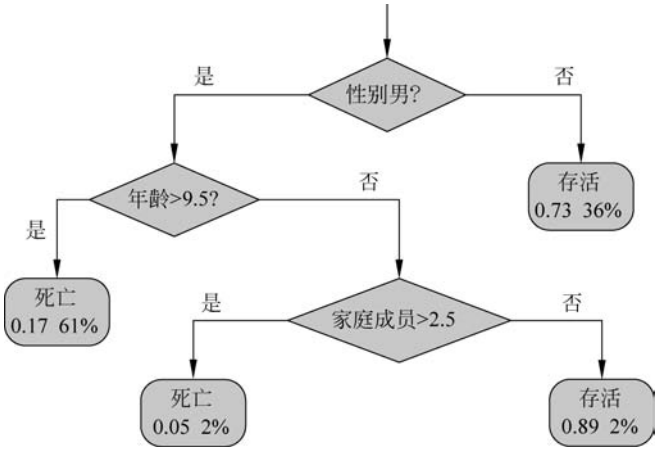


图 3-9　决策树

4. 现代机器学习的成型时期

1990 年，Schapire 最先构造出一种多项式级的算法，这就是最初的提升(Boosting)算法。一年后，Freund 提出了一种效率更高的 Boosting 算法。但是这两种算法存在共同的实践上的缺陷，那就是都要求事先知道弱学习算法学习正确的下限。1995 年，Freund 和 Schapire 改进了 Boosting 算法，提出了 AdaBoost(Adaptive Boosting)算法，该算法的效率和 Freund 于 1991 年提出的 Boosting 算法几乎相同，但不需要任何关于弱学习器的先验知识，因而更容易应用到实际问题当中。

Boosting 是一种用来提高弱分类算法准确度的框架算法，这种方法通过构造一个预测函数系列，然后以一定的方式组合成一个预测函数。Boosting 算法通过对样本集的操作获得样本子集，然后用弱分类算法在样本子集上训练生成一系列的基分类器，如图 3-10 所示。

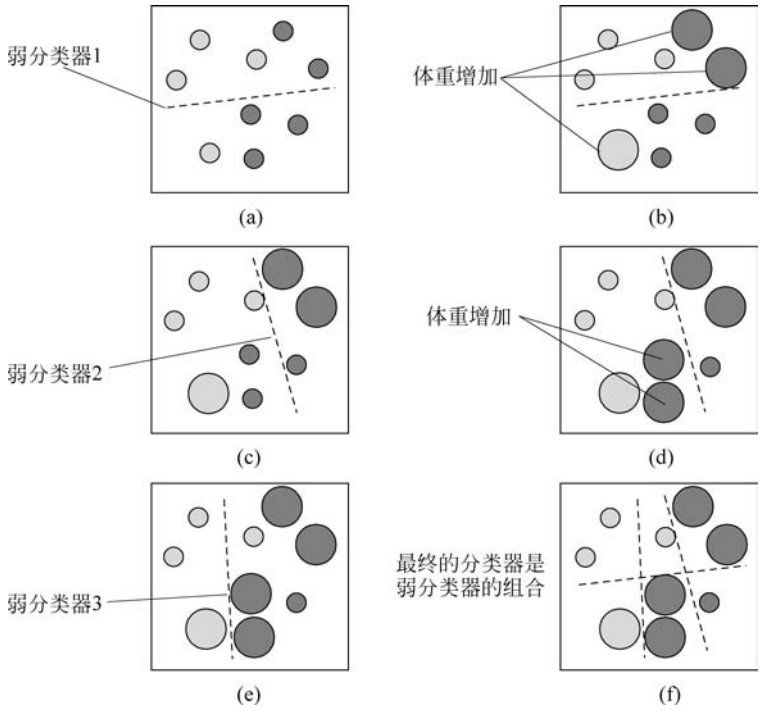


图 3-10　Boosting 算法

支持向量机(Support Vector Machine,SVM)于 1964 年提出,是机器学习领域的另一大重要突破,该算法具有非常强大的理论地位和实证结果。那一段时间机器学习研究也分为神经网络(Neural Network,NN)和支持向量机(SVM)两派。然而,在 2000 年左右提出带核函数的支持向量机后,SVM 在许多以前由 NN 完成的任务中获得了更好的效果,如图 3-11 所示。此外,SVM 相对于神经网络(NN)还能利用所有关于凸优化、泛化边际理论和核函数的深厚知识。因此,SVM 可以从不同的学科中大力推动理论和实践的改进。

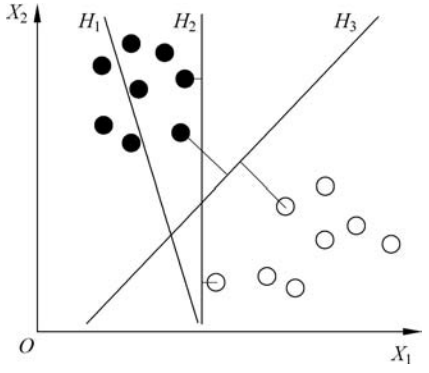


图 3-11　支持向量机 SVM

与 SVM 相比,这一时期神经网络(NN)遭受到又一个质疑,通过 1991 年和 2001 年 Hochreiter 等人的研究,表明在应用 BP 算法学习时,NN 神经元饱和后会出现梯度损失(Gradient Loss)的情况。简单地说,在一定数量的 epochs(样本数量/批量大小)训练后,神

经网络模型会产生过拟合现象。这一时期，因为神经网络模型相比 SVM 更容易过拟合，导致神经网络处于劣势。

2001 年，布雷曼博士提出“随机森林”的概念，其基本单元是决策树，是通过集成学习的思想将多棵树集成的一种算法。随机森林的本质属于机器学习的一大分支——集成学习（Ensemble Learning）方法。随机森林的名称中有两个关键词；一个是“随机”，一个就是“森林”。“森林”很好理解，一棵叫作树，那么成百上千棵树就可以叫作森林，这样的比喻还是很贴切的，其实这也是随机森林的主要思想——集成思想的体现。

5. 爆发时期

2006 年，神经网络研究领域的泰斗杰弗里·辛顿提出了深度学习（Deep Learning）算法，使神经网络的能力大大提高，挑战支持向量机。杰弗里·辛顿和他的学生鲁斯兰·萨拉赫胡迪诺夫在顶尖学术刊物 *Science* 上发表了一篇文章，开启了深度学习在学术界和工业界的浪潮。

从几十年的历史可以看出，机器学习的发展并不是一帆风顺的，也经历了螺旋式上升的过程，成就与坎坷并存。机器学习的发展诠释了多学科交叉的重要性和必要性，各领域研究学者的成果共同促成了今天人工智能的空前繁荣。

3.1.3 机器学习应用领域

机器学习经历了几十年的发展，虽然并不能与人类的智能相比，却在日常生活中得到了广泛的应用。

（1）银行、零售和电信。例如，潜在客户和合作伙伴，客户满意度指数（基于关系、交易、营销活动等），欺诈、浪费和滥用索赔，预测信用风险和信誉，营销活动的有效性和影响因素，交叉销售和建议，联络中心帮助客服代表在与客户的通话中获取相关数据。

（2）医疗保健和生命科学。例如，扫描、筛选和生物识别，基于混合成分的药物，基于症状、患者记录和实验室报告的诊断和补救，根据药物、患者、地理位置、气候条件、过往病史、食物摄入等数据的 AECP（不良事件病例处理）情景。

（3）一般日常应用。例如，文字或语音书写识别，调试、故障排除和解决方案向导，过滤垃圾邮件，短信和邮件分类或建议，支持问题并丰富 KeDB（知识错误数据库），朋友和同事推荐，无人驾驶，通过构建人工智能和算法，图像处理等。

（4）安全应用。例如，手写、签名、指纹、虹膜/视网膜识别和验证，人脸识别，DNA 模式匹配。

综上所述，机器学习非常适用与以下几种场景：

- 人类不能手动编程。
- 人类不能很好地定义这个问题的解决方案是什么。
- 人类不能做到的需要极度快速决策的系统。
- 大规模个性化服务系统。

如图 3-12 所示，对于人类的头脑来说，反复数十亿次的不间断处理数据，必然是会感到厌倦的，这就是机器学习算法发挥关键作用的地方。

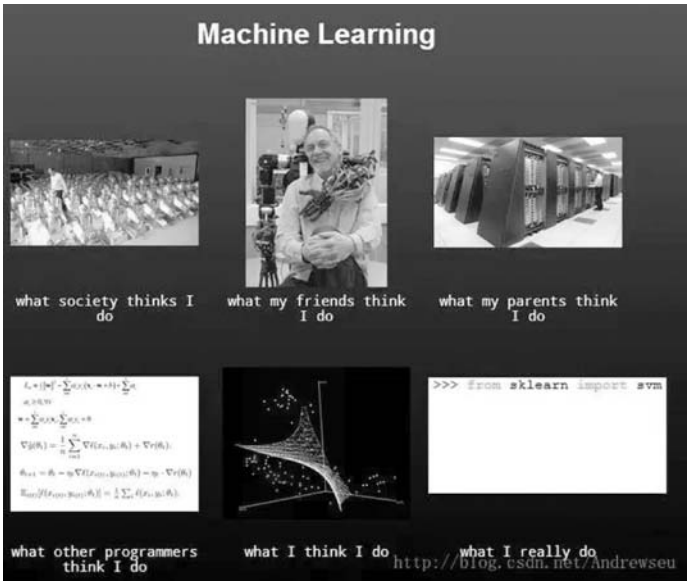


图 3-12　机器学习适用场景

3.2　机器学习进阶

如果把机器学习比作一个人，那么到目前只是认识了这个人，但对他的性格以及特质还是很模糊，并不了解。接下来，将进一步深入了解机器学习的基础知识，进一步地观察机器学习到底长什么样以及特质，并学习一些专业的术语，能够更准确和专业地评价和讨论机器学习。

3.2.1　机器学习种类


视频讲解

实现机器学习的方法多种多样，按照不同的应用领域和案例，使用的机器学习的方法也不尽相同。

1. 按功能分类

从功能上来说，机器学习的功能大致可分为回归、分类和聚类。初次听到这几个词会比较抽象，可以通过在学校中的例子来想象一下。

1）回归

关于回归，假设某同学所在的班级每周 x 都会进行一次考试，每次考试都能得到自己所在班级的一个排名 y。那么经历了 n 次考试后（每次考试未必是相邻的周），能够根据每次的考试名次大致画出这个同学名次波动的曲线。根据这条曲线，也就基本可以推测该同学在过去或未来某一次考试的名次。这条曲线生成的过程就属于回归的拟合过程。

回归是指有无限个可能的问题，根据已知变量去预测的是一个连续的、逼近的变量。比如房价的预测、明日气温的预测。在实际的回归案例中，已知变量也是连续的值，会比较复杂，具体操作会在后续章节讲解，现在只需要知道回归问题预测的是一个连续的值就可以了。

2）分类

关于分类，是事物所有状态的可能性的集合。比如对于生命状态的分类包括生和死，对产品合格状态的分类包括合格与不合格，对于动物的分类包括猫、狗和其他物种，对于学生毕业就业的情况可分为就业与未就业。每当讨论起某同学毕业一定能找一个好工作，这样的一个推论，往往是根据该同学许多因素得来的，例如，学习成绩是否良好，是否是班委，是否在学校有兼职，是否有大赛获奖情况等因素。但是这些因素所起到的作用一定有轻重之分。因此，根据毕业生的在校情况给每个因素赋予一定的轻重比例，这个过程就是分类的拟合过程。根据这些因素比例，就可以轻松地推测在校生未来是否能够就业。当然，没人能够神机妙算，预测的只是每一个类别的概率，而不是实际数据。

分类是指有限个可能的问题，预测的是一个离散的、明确的变量。比如给出一张图片，去判断是 T 恤、是裤子，还是其他的种类。同样，在实际的分类问题中，已知变量有可能是连续的值，也有可能是离散的（表示类别）的值，人们只需要知道分类问题预测的是一个概率就可以了。

3）聚类

关于聚类，和分类有一些相似，都是把一个数据集中的数据分成不同的类别。但是这些数据往往没有明确的标签。在学校每年一度的运动会上，各个班级的同学在操场上按不同的区域集合，当大家在休息时，各班级的同学多少会聚集。虽然散开的同学身上并没有标注班级的标签，但可以根据同学相互之间的距离和聚集程度大体区分出不同的班级。

聚类需要从没有标签的一组输入向量中寻找数据的模型和规律，在数据中心发现彼此类似的样本所聚成的簇。

2. 按方式分类

人工智能是模仿人在某一方面的能力，从机器学习的方面来讲，就是能像人一样预测数值、预测可能性。和人类一样，机器学习生来并不具备这些能力，机器学习准确的推测是建立在通过大量已知数据的分析基础上的，这个过程称为“学习”或者训练。机器学习的学习方式大体分为 3 类：监督学习、无监督学习和弱监督学习。

1）监督学习

最常见的监督学习，就是平时学习然后参加考试的过程。中学阶段通过大量的习题训练，学习解题方法，为的是能够在高考的时候取得好的成绩，但是高考的题目是之前没有见过的，但是这并不说明高考的题目是不能做出来的，学生可以通过对之前做过的习题的分析，找到解题的方法。

监督学习的原理和上面的例子差不多，利用一些已知的数据训练机器（做习题），然后机器分析数据，找到内在联系（学习解题方法），从而对未知的数据进行做出判断和决策（做高考题）。

监督学习的学习数据既有特征（Feature），也有标签（Label）。例如学生就业，数据中既有学生的平时表现数据，也有学生最后的就业数据，利用这些数据，机器学习就生成了预测模型。其中的标签，也可以理解为一种反馈。例如，学生在学习过程中，所做的习题没有参考答案来给出反馈，那做题就是出于瞎猜的状态，学习效果也会受到影响。

监督学习首先要通过学习已知数据，然后才可以完成预测和分类。

2）无监督学习

前面所说的聚类过程就属于典型的无监督学习。相对于有监督学习，无监督学习是一类比较困难的问题，其学习数据没有标签，同时特征往往不够明确，按照一定的偏好，也能够大致将所有的数据映射到多个不同标签（分类）。所谓按照一定的偏好，是比如特征空间距离最近等人们认为属于一类的事物应具有的一些特点来进行分类。

3）弱监督学习

监督学习可以更精确地拟合现实中的模型，但监督学习的缺点也很明显。首先，监督学习要求有大量的训练数据；同时，这些训练数据必须有标签。然而，数据的标记工作通常由人工完成，这就造成人的工作量大，同时也难以保证标签的正确性。弱监督学习便是为了应对各种数据标签问题提出的解决方法。弱监督通常分为 3 种类型：不完全监督、不确切监督、不准确监督。

不完全监督，指的是训练数据只有部分是带有标签的，同时大量数据是没有被标注过的。这是最常见的由于标注成本过高而导致无法获得完全的强监督信号的情况。例如，聘请领域专家直接给大量数据添加标签的成本就相当高。另外，在为医学影像研究构建大型数据集时，放射科医生可能不愿意标记数据增加他们的工作量。根据以往的经验，由于医生对于数据科学的了解往往不够深入，有许多数据的标注结果（例如，为分割任务框定的病灶轮廓）是无法使用的，从而产生了很多实际上缺少有效标记的训练样本。

不确切监督，即训练样本只有粗粒度的标签。例如，针对一幅图片，只拥有对整张图片的类别标注，而对于图片中的各个实体（instance）则没有标注的监督信息。例如，对一张肺部 X 光图片进行分类时，只知道某张图片是肺炎患者的肺部图片，却并不知道具体图片中哪个部位出现问题，对应说明了该图片的主人患有肺炎。

不准确监督，即给定的标签并不总是真值。出现这种情况的原因有很多，例如，标注人员自身水平有限、标注过程粗心、标注难度较大。在标签有噪声的条件下进行学习就是一个典型的不准确学习的情况。

3.2.2　基本术语

机器学习的机制是地地道道的人类思维，非常好理解。在讨论机器学习的机制前，先统一基本术语，这是进行沟通和理解问题的基础。掌握了这些术语，就不再是一个简单的使用者，而是跨入了人工智能专业技术的大门。

1. 数据集

数据是机器学习的原材料，是机器学习产生“智能”的源泉。假如要让机器学习学会判断一个西瓜是否成熟，那就需要提供一组关于西瓜的数据。例如，西瓜的颜色、西瓜的尺寸、西瓜的重量、西瓜的根蒂形状、敲击声如何等。如果只提供一两个西瓜的数据，机器学习很难正确地学会挑熟西瓜。因为凡事都有例外，这个例外的数量还可能不少。所以用于机器学习的西瓜的数量必须很大，如一地摊西瓜的数据，或者是一大车西瓜的数据。这种由多数量样本组成的数据，称为“数据集”。

在这个数据集中，反映西瓜特点的属性，如西瓜的颜色、西瓜的大小等称为属性或特征。对于西瓜重量这个特征值来说，可以从几克到几千克，是一个连续值，对于这种具有连续值的数据，称为“数值数据”；而对于西瓜的颜色这个特征值来说，取值为“青绿”或者“乌黑”，

是由好几个分类组成的，是不连续的数据，称为“分类数据”。

在西瓜的数据集中，每一个西瓜的特点是通过（颜色，尺寸，重量，…，形状）多个特征值构成的，这些多特征组合称为“特征向量”。

2. 模型

通过西瓜的特征，经过思考，就可以辨别西瓜的成熟度。看似简单的一个事情，思考一下，是怎样实现的。首先人的眼睛会把西瓜的样子传入大脑，人的手会把西瓜的重量传入大脑，最终这些信号进入复杂的神经连接，最终在传导人的嘴上，说出了结果。正是这样一个复杂的生理结构让人能够做出思考和判断。机器学习也是同样的道理，任何一个机器学习也应该有这样一个“生理结构”，而这样一个生理结构称为“模型”。模型的构建是机器学习非常重要的一部分。对于初学者来说，不要因此而却步，在很多情况下，只需要会用某个模型就可以了。就如同消费者都可以根据自己的需要买到性能不错的手机，但并不需要自己会制造手机。

3. 训练

生理结构是人类智能的基础，可是只有生理基础，人类智能也是不完善的。没有人天生就会挑瓜，这是经历过无数次成功和失败的亲身实践后才可能具备的技能。这样一个过程就是机器学习中的“学习”。一个建立好的模型就如同一个新生的婴儿，只是有了最基本的生理结构，如果不经过长期的学习，其智能也不会有任何提高。让建立好的模型学习大量数据的过程称为“模型训练”，而用于训练的那部分数据集，称为“训练集”。

4. 预测

在训练过程中，预测结果会对模型起到评价作用，使用预测结果与实际结果的差距来调整模型的参数来改善模型。比如我们挑了一个瓜，认为是熟的，但切开之后却是生的。这个结果对我们会起到反馈作用，让我们下一次遇到类似情况时避免犯错。如果把训练比作我们平时的学习，而预测就相当于做作业。作业会让我们知道哪些知识点需要加强学习。

5. 测试

同训练集一样，一组用于测试的大量数据构成的数据集，称为“测试集”。在训练结束后，将测试数据输入训练好的模型，根据测试集的正确率来衡量模型性能的好坏。这里需要注意的是，测试集中的数据和训练集是完全不同的，就好比考试的题目不应该出现原题一样。如果用训练集做测试，会发现正确率很高，但这是一种虚假的结果，并不能反映模型训练得好不好。


视频讲解

3.2.3 机器学习的流程

机器学习的一般流程其实和传统的软件开发的流程有些不同，不必将精力放在编程的流程上，而是要理解每一步的具体意义。在实际的应用过程中，对待不同的案例流程和表述会有一定的差别，总体上来说机器学习的流程包括分析案例、数据获取、模型训练和模型验证这 4 个过程。

1. 分析案例

机器学习的第一步首先是要理解实际问题，把现实问题抽象为机器学习能处理的数据问题。鸢尾花有不同的品种，可以通过肉眼直观地看出不同品种的区别。如果要让机器学习能够区分不同的品种，就需要把花瓣样子抽象为数据。在这个案例中，可以把鸢尾花的花

瓣长度和花瓣宽度的数据作为要处理的数据。

门卫工作人员可以识别出不同的访客，是因为每个人都有不同的面目特征。如果要让机器学习也能够进行人脸识别，就需要把人脸的图像抽象为数据。由于人脸的特征过于复杂，去测量统计每个人眼睛的形状，五官的距离是不现实而且不准确的。在这种情况下，不妨把人脸图像量化为数字图像，把图像中的每一个像素都作为处理的数据。

2. 数据获取

第二步就是准备机器学习所需要的数据集了，包括获取原始数据以及从原始数据中经过特征工程从中提取训练数据、测试数据。在实际应用中，只靠个人去准备数据是相当费时费力的工作，因为少量的数据并不能很好地支撑机器学习。机器学习在当代之所以能再次发展起来，其中重要的一个原因是几十年来相关案例的数据收集比较完善。不少科研机构和公益组织准备了大量的数据集供人们学习和使用。常见的开源数据集网站如表 3-2 所示。

表 3-2　开源数据集汇总

名　　称	简　　介	网　　址
UCI	UCI 是加州大学欧文分校所维护的一个数据集(库)，里面包含 373 个数据集，包括分类、回归等多种类型	http://archive.ics.uci.edu/ml/index.php
GoogleTrends	GoogleTrends 数据集的来源主要是互联网，具有很强的时效性，社会性，更具有应用价值。数据主要以 csv 表格文件保存	http://googletrends.github.io/data/
Kaggle	Kaggle 本身是为开发商和数据科学家提供举办机器学习竞赛、托管数据库、编写和分享代码的平台，里面的数据覆盖了分类、回归、排名、推荐系统以及图像分析等各个非常实用的领域	https://www.kaggle.com/
AWS 公用数据集	AWS 公用数据集需要通过自身的 API 访问，包含了人类基因组项目、Common Crawl 网页语料库、维基百科数据和 Google BooksNgrams 等形形色色的数据集	
Imagenet	Imagenet 是图像领域最出名的数据集之一，该数据集有 1400 万张图片，涵盖有丰富的类别，带标注数据也超过百万，这使得该数据集在图像处理，定位，检测等研究工作中占据很大的地盘，其机会成为目前深度学习图像领域算法性能检验的标准数据库	http://www.mhylpt.com/
MNIST	MNIST 是一个手写数字数据库，它有 60 000 个训练样本集和 10 000 个测试样本集，每个样本图像的宽高为 28×28	http://yann.lecun.com/exdb/mnist/

这些数据集提供的数据特征比较丰富，在学习和应用过程中，应该根据实际情况去分析选择自己所需要的特征。如图 3-13 和图 3-14 所示，选择样本数据时，一般采用随机抽取的方式组成自己的训练集和测试集，因为随机组成的数据更能反映一般规律。

图 3-13　数据集图片

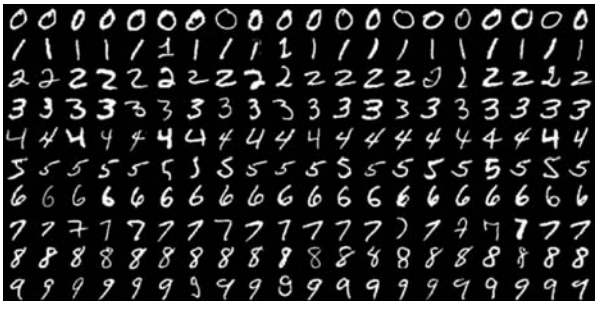


图 3-14　MNIST 数据集

3. 模型训练

在准备好数据之后，要首先选择一个适合自己的模型，不同的模型在解决不同的问题上性能会有比较大的差距。之后便是将准备好的数据交给模型来训练，通过迭代使模型最终收敛。在训练的过程中，需要根据训练的精度误差以及训练时间来调整相应的参数，使得模型更适合实际案例。

4. 模型验证

一旦训练完毕，就要对得到的模型进行评估。此时，早前选好的测试集就派上用场了。在评估中，使用之前从未使用过的数据来测试模型，得到输出并与正确的判定结果对比。模型验证主要用来衡量训练后模型的性能。

视频讲解

3.3　线性模型

线性模型(Linear Model)蕴含了机器学习中一些重要的基本思想。线性模型形式简单、易于建模，具有很好的解释性。线性模型尝试通过属性的线性组合来进行描述和预测。

$$f(X)=w1*x1+w2*x2+\cdots+wn*xn+b$$

其中，x_i 是第 i 个属性的取值；w_i 是属性的系数。

在下面的案例中，重点学习线性模型是如何实现简单的应用，变成为人所用的工具，感受一下成为“先知”的快感。对于具体的方法，不需要纠结如何实现，明白原理和思路才是重要的。

3.3.1　预测工资——线性回归

1. 训练流程

1）场景说明

通过员工工作年限与工资的对应关系表，找出二者之间的关系，并预测在指定的年限时，工资会有多少。

可以看出，这是一个用工作年限预测工资的简单线性回归问题。下面就按照最简单的流程来解决这个问题。

2）确定数据

观察数据是进行数据分析等机器学习过程的第一步。这一步的主要目的是对训练数据有初步的认识。比如观察数据的特征含义，确定哪些特征是有效的，哪些特征是无效的，数据是否完整等。这一步能够帮助我们对数据有一个大致的理解。

员工的工资信息如表 3-3 所示，每一列代表数据的一个特征，分别是“姓名”“年限”“级别”“工资”。在这些特征中，“姓名”一般不会影响到工资收入，可以将“姓名”作为无效的特征，不纳入训练的数据当中。因此这个数据集的特征向量即为{年限，级别，工资}。“工资”是结果，而“年限”和“级别”两个因素共同决定了“工资”。“工资”之所以难以预测，是因为工资并不是根据某个固定的公式计算出来的，其中也包含了复杂的“人”的因素，比如员工的为人处世、老板是否赏识等。

表 3-3　年限和工资数据

姓　　名	年　　限	级　　别	实际工资/元
蹇嘉怡	5	1	5500
焉从丹	3	7	5500
问德曜	8	2	10 000
经茂彦	1.5	5	4500
仰雅旋	10	6	13 000
来囡囡	3	3	5500
浮彬郁	7	3	8500
夷三姗	5	4	7000
和云臻	1	2	4000
桥琪华	8	5	11 500
犹　青	3	4	4000
宿雪萍	2	3	3500
莱　颀	4	1	4500
池晶辉	4	4	6000
星迎蕾	3	2	3500
百淑贞	1	3	2500

续表

姓　　名	年　　限	级　　别	实际工资/元
凭嘉福	4	2	5000
庆月灵	5	2	6000
麴　茵	3	4	5000
仉云水	7	5	9500
明　甘	6	5	8500

以上数据保存为 csv 表格文件，使用 Python 的 pandas 库读取以上数据的代码如下：

```
#工作年限、级别与工资数据(csv 文件)
csv_data = 'salary.csv'
#读入 dataframe
df = pandas.read_csv(StringIO(csv_data))
print(df)
```

3）确定模型

通过线性模型来预测一下年限和工资的关系。为了简单理解，首先考虑“年限”和“工资”两者之间的关系。假定工资表示为 y，年限表示为 x，两者符合线性模型，那么这个模型就可以设为 $y=ax+b$。使用二维坐标来显示年限和工资数据如图 3-15 所示，年限和工资基本呈现线性增长的形状。

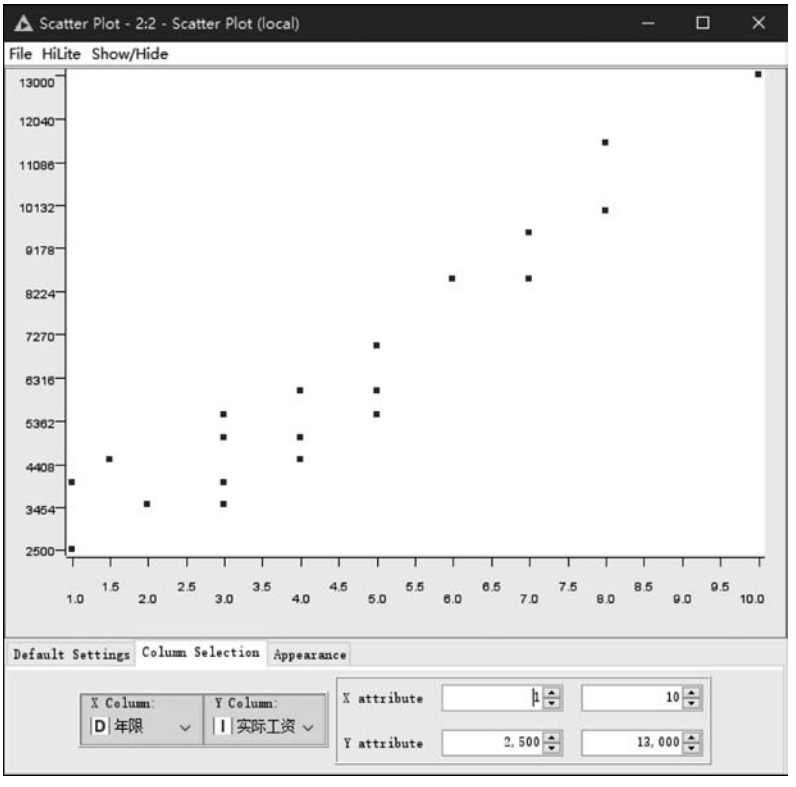


图 3-15　工资年限数据样本

使用 Python 的 sklearn 库建立模型的代码如下：

```
#建立线性回归模型
regr = linear_model.LinearRegression()
```

4）训练模型

在确定了模型之后，就可以使用现有的数据来训练模型。训练的过程实际上就是调整参数 a、b 的过程，这个过程叫作拟合。训练的目标就是确定 $y=ax+b$，使这条直线最大限度地接近这些散点。

可以想象，如果员工的数据非常少，那么参数 a 和 b 计算的结果是不准确的，随着员工的数据增多，也就是训练集变大，参数 a、b 的计算结果将越来越精确。当然，随着训练的次数不断增多，a、b 的值会有一个精确的极限，当它们不能够再被精确时，则被认为这个模型已经训练得“足够”好了，这个结果叫作“收敛”。

如图 3-16 所示，一开始只用前 3 个数据来训练（图中红框标记的点），这时拟合的直线为 $a=947.3684$，$b=1947.3684$（图中红色的直线）。在图中可以直观地看出，红色的这条线并不能最大限度地贴合各个样本数据，认为在数据量少时，结果并不精确。当使用所有的 22 个数据进行训练时，生成的黑色直线明显要优于红色的直线。此时 $a=1083.073$，$b=1511.0797$。图中这条线就是线性回归的结果，其实际意义代表在已知的工作年限下，对应的高度就是预测的收入。不同迭代次数下的拟合结果如图 3-17 所示。

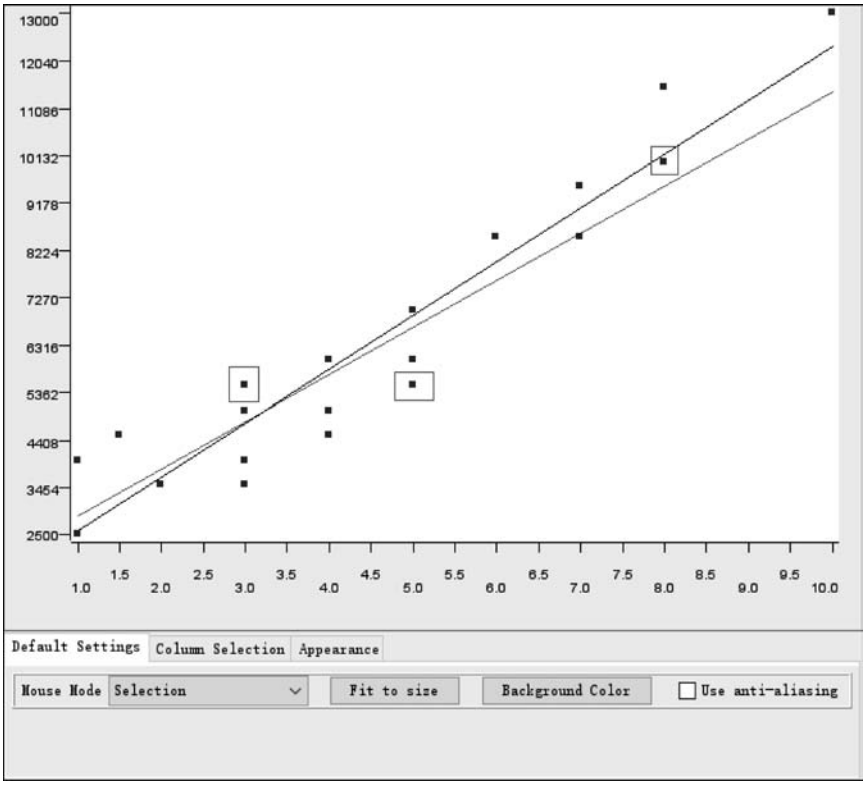
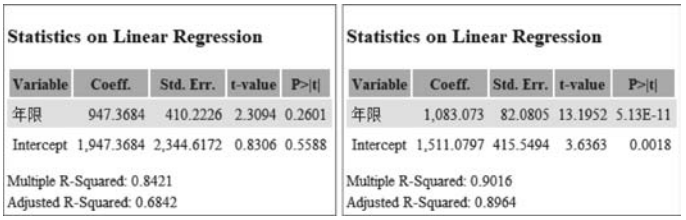


图 3-16　不同迭代次数下的工资年限回归线（见彩插）



Statistics on Linear Regression

Variable	Coeff.	Std. Err.	t-value	P>|t|
年限	947.3684	410.2226	2.3094	0.2601
Intercept	1,947.3684	2,344.6172	0.8306	0.5588

Multiple R-Squared: 0.8421
Adjusted R-Squared: 0.6842

Statistics on Linear Regression

Variable	Coeff.	Std. Err.	t-value	P>|t|
年限	1,083.073	82.0805	13.1952	5.13E-11
Intercept	1,511.0797	415.5494	3.6363	0.0018

Multiple R-Squared: 0.9016
Adjusted R-Squared: 0.8964

图 3-17　不同迭代次数下的拟合结果

使用 Python 的 sklearn 库训练模型的代码如下：

```
#填入数据并训练
regr.fit(df['年限'].reshape(-1, 1), df['实际工资']) # 注意此处.reshape(-1, 1),因为X是一维的!
#得到直线的斜率、截距
a, b = regr.coef_, regr.intercept_
print(a,b)
```

5）让预测更精确

通过上面的实验可以看到，实际预测的工资和真实工资总是有或大或小的差距，这条线只是代表了整体预测的误差最小的情况。那么使预测更加精确就是训练模型并进行调优的目标。

在上面的模型中，只使用了一个特征值{年限}。这种使用一个特征去拟合另一个特征的回归，称为一元线性回归。在实际的数据中，还存在另一个特征“级别”，该特征也会对“工资”产生影响，因此应该将此特征也纳入训练的过程中，将特征向量的尺度由一元变为二元的{年限，级别}。这种由多个特征去拟合另一个特征的回归，称为多元线性回归，此时的模型就变为 $y=ax1+bx2+c$。利用这个新的模型，重新训练数据并观察结果。

从图 3-18 可以看出，二维特征向量加上一个结果特征构成了三维空间中的点，而空间中的平面则是二元线性回归拟合的平面，平面上的任意一点就是该点对应的年限和级别时所预测的工资。从上面可以看出，随着线性回归特征向量尺度的增加，模型也会跟着变得复杂，一般来说，训练的结果也会变得更好。

一维特征向量拟合的是一条直线，二维特征向量拟合的是一个平面，那么三维、四维拟合的结果会是怎样的？对于高维向量，并不能用三维空间坐标表示出来，这个理论的高维平面称为超平面。在实际很多应用中，对于事物的描述往往通过多个特征来描述，训练数据的特征向量也就基本都是高维特征向量了。

2. 训练原理

模型通过训练，实现了比较准确的预测功能。下面将介绍模型是怎样训练才能达到这样的结果。在这里，大家不必过于担心，我们不会陷入数学的陷阱，只需要了解训练的原理就可以了。

想象一下，一个婴儿是怎样能够学会辨别猫还是狗呢？首先，他必须接触大量猫和狗的图片，其次，每一次对猫或狗的辨认一定需要家长给予反馈。辨别错了，家长会表现得不开

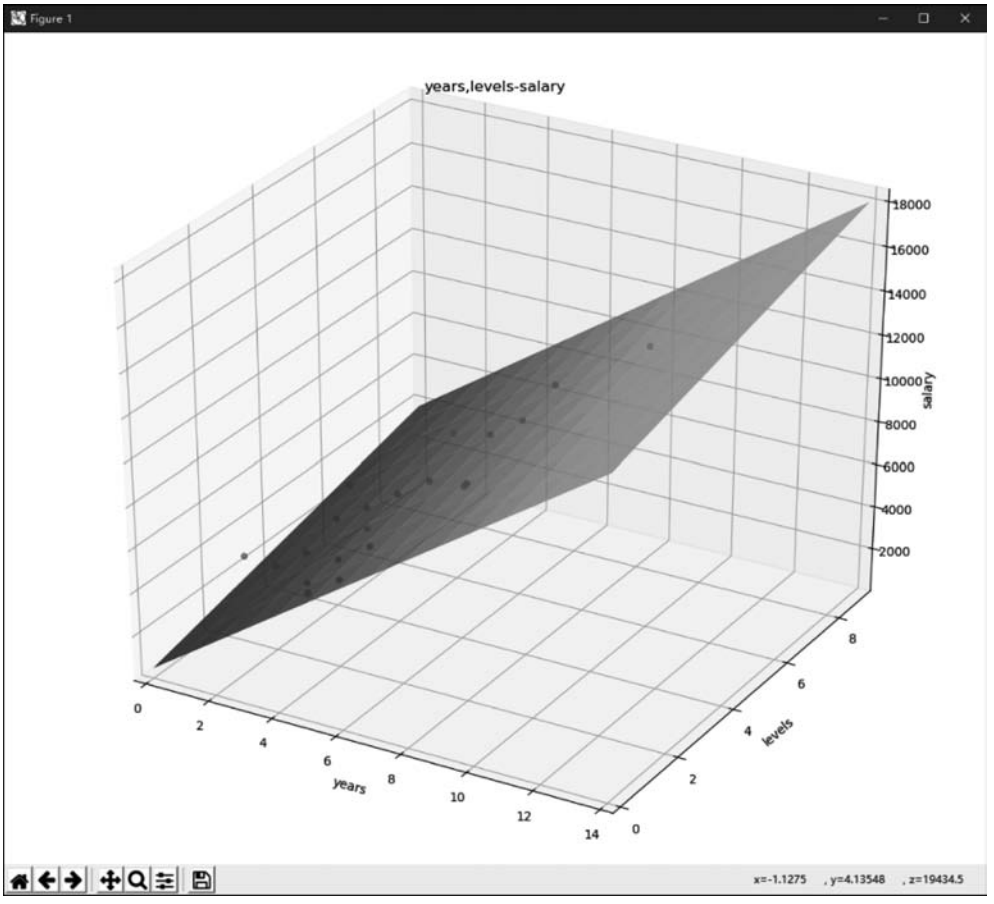


图 3-18　二元拟合平面

心，语气生硬；而辨别正确，家长则会表现得开心，并给予奖励。正是在这样的反馈当中，婴儿的辨识能力不断加强。在机器学习中扮演这个家长角色的就是损失函数。损失函数是用来估量模型的预测值与真实值的不一致程度，它是一个非负实值函数。损失函数越小，模型的稳健性就越好。训练的过程就是模型通过不断地迭代调整各个参数，使得损失函数达到一个相对最小的状态。

1）梯度

接下来的问题是，怎样调整模型参数使得损失函数不断变小呢？到这了个层面，问题其实更接近于数学编程问题，只需要理解方法即可。使得损失函数最小化的方法一般为梯度下降法。可以把梯度想象为表示一个曲线或曲面上某一点的陡峭程度，如图 3-19 所示，分别在紫点和红点的地方做一条切线可以发现，两条切线的方向不同，切线的倾斜角度不同。紫色点位置的切线斜率为负，称为负

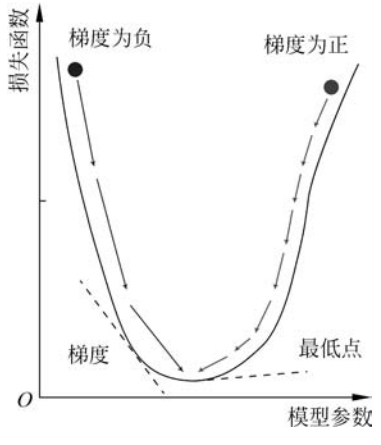


图 3-19　损失函数梯度曲线（见彩插）

梯度；红色点位置的切线斜率为正，称为正梯度。这里的正负只表示为方向，并不表示大小，所以红色点位置的梯度会更小一些。有这样一个规律不难看出，如果人走在一条崎岖的道路上，先走下坡路，后走上坡路，那么在这段路中间一定存在一个最低的点。即在梯度分别为正负的两个点之间，一定存在一个梯度为 0 的点。如果模型的损失函数是一元的，那么就可以把模型参数和损失函数表示为图 3-19 中所示的样子，目标就是在这条曲线中找到位置最低的那个点，这个点就是损失函数的最小值点，一般认为此时模型达到了收敛。

同样，如果损失函数是二元的，则可以把模型参数和损失函数拟合为一个曲面。如图 3-20 所示为珠穆朗玛峰的地形曲面，颜色越深表示地形越低，颜色越浅表示地形越高。因此颜色变化幅度大的地方，就是梯度大的地方，目标就是从这个曲面中找到最低的那个点。

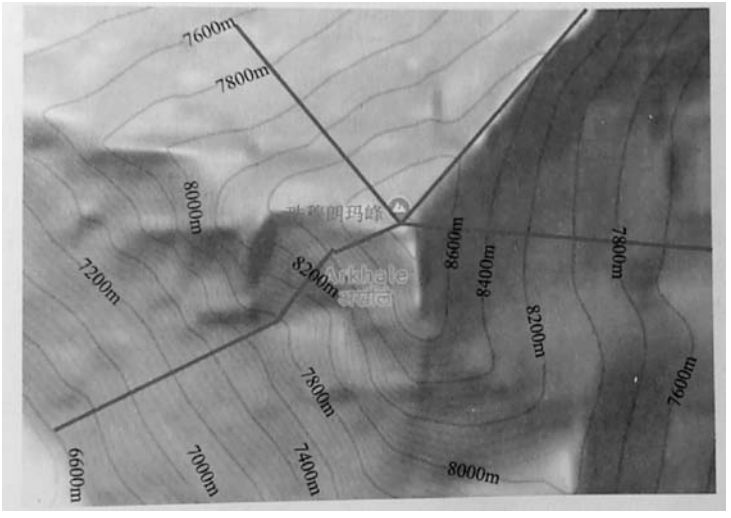


图 3-20　二元梯度曲面

2）梯度下降

在图 3-19 中的梯度曲线中，从紫色的点走到最低的那个点，可以利用下面公式：

下一个位置＝当前位置－学习率×梯度

当初始位置在左侧时，往右走一段距离（学习率），看看当前位置是不是比原来的位置更低，如果是，就继续往右走；当下一步跨过最低点时就会发现，当前的位置没有更低反而升高时，这就说明走过了，需要反过来再往回走。这样不断循环，最终就会找到最低点。

3）学习率

在上面的公式中可以看出，使用梯度下降法最重要的一个因素是学习率。如图 3-21 所示，设置不同的学习率。如果学习率定得太高，步子迈得太大，好处是可以走得很快，但总是会在最低点附近跨来跨去，最终找到的最小值离实际的最小值误差会比较大；如果学习率定得太低，步子迈得太小，会更容易接近实际的最小值，但是速度会变慢、效率低。如何设置

学习率则考验机器学习的运用能力和经验。

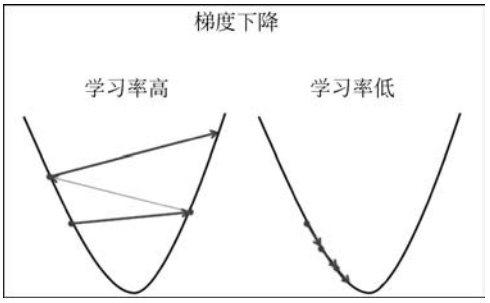


图 3-21　不同的学习率

4）过拟合问题

再次回到工作年限与工资收入的关系这个问题，只用一条直线来拟合年限与工资的关系，如果使用一条曲线来拟合，那么这条曲线会更加贴合每一个样本数据，也就更加精确了，如图 3-22 所示。

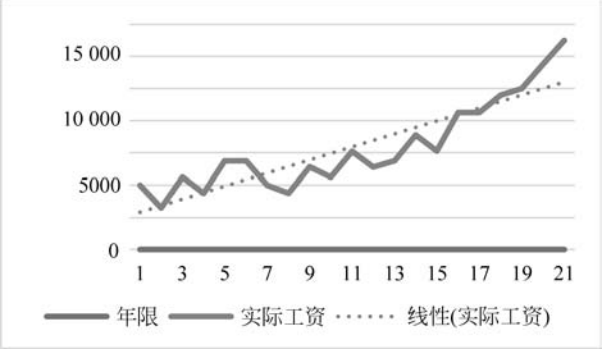


图 3-22　一次多项式拟合曲线（见彩插）

$y=ax+b$ 属于一次多项式，把次数增加为二次多项式，模型就变为 $y=ax^2+bx+c$，此时拟合的预测线就由直线变为图 3-23 中的曲线。

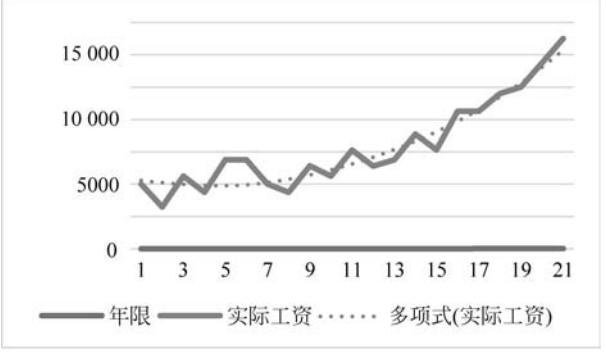


图 3-23　二次多项式拟合曲线（见彩插）

继续把模型变为三次多项式 $y=ax^3+bx^2+cx+d$，此时预测线就变为图 3-24 中的曲线。

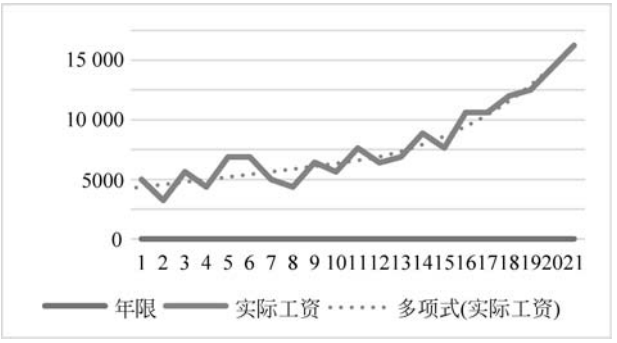


图 3-24　三次多项式拟合曲线（见彩插）

以此类推，不断把模型变为四次、五次、六次，观察如图 3-25～图 3-27 所示的曲线在图中的变化情况。

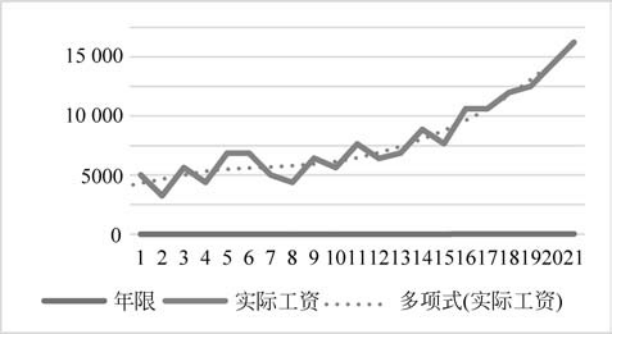


图 3-25　四次多项式拟合曲线（见彩插）

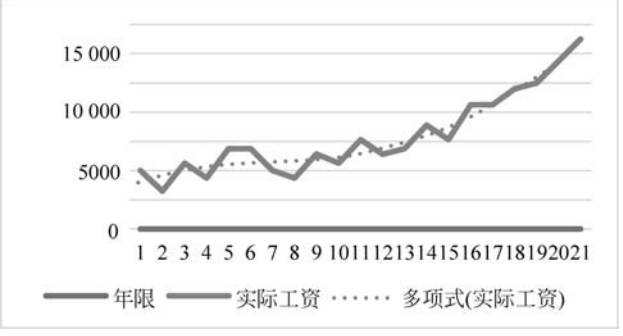


图 3-26　五次多项式拟合曲线（见彩插）

可以看出，随着模型次数的增加，预测线会拟合得越来越好，越来越贴近实际的采样点，但这并不能说明模型越来越好。

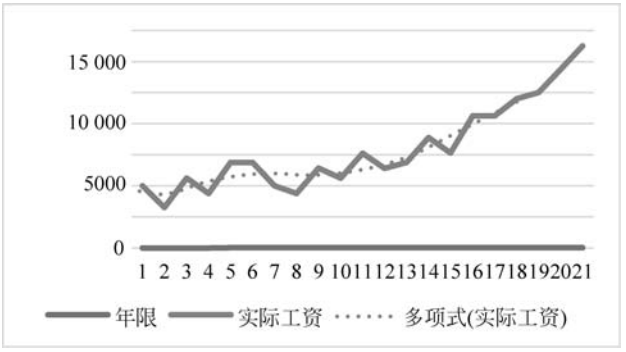


图 3-27　六次多项式拟合曲线(见彩插)

因为现实中样本的特征数据并不是完全精确的,其中会有很多干扰因素,例如,有的老板和某些员工更合拍,发的工资就会更高,在这种情况下,年限这个因素相比就是次要因素。在年限-工资这个模型中,模型拟合得越好,就代表模型把年限和工资建立起更强的联系,越把年限因素看成绝对因素,一些特殊的、不符合规律的样本点越会被采纳。就像是书呆子,只会考试某种试题,稍微一变就不会了。

过拟合就是模型完美地或者很好地拟合了数据集中的有效数据,同时也很好地拟合了数据集中的错误数据,但是此模型很可能不能很好地用来预测数据集的其他部分。如图 3-28 所示,如果拟合的曲线完全符合样本点,不难看出对于大部分位置的预测都是离谱的,这就属于严重的过拟合。

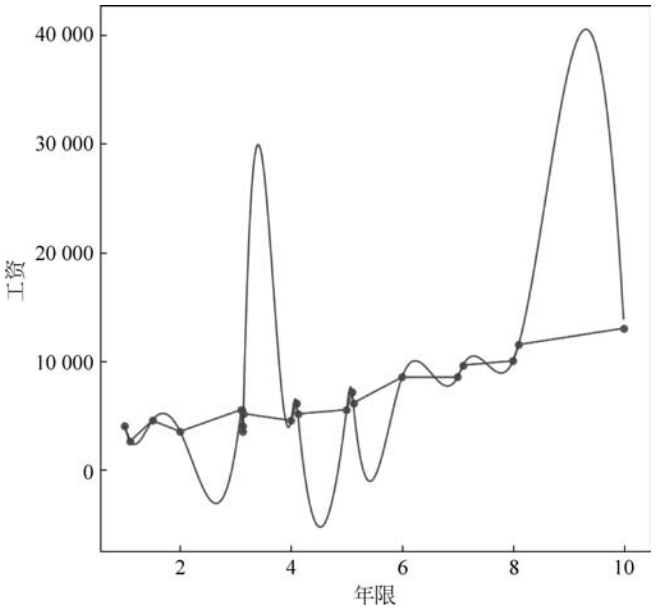


图 3-28　过拟合的极端情况(见彩插)

3.3.2 泰坦尼克号生存预测——逻辑回归

1. 训练流程

1）场景说明

泰坦尼克号是一艘英国皇家邮轮，如图 3-29 所示。是当时全世界最大的海上船舶。1912 年 4 月 10 日，泰坦尼克号展开首航，乘客身份各种各样，他们有不同的年龄，来自不同的国家，拥有不同的财富，有着不同的家庭成员，他们的票价和所在的船舱等级也不同。在 4 月 14 日凌晨，它在中途岛碰撞冰山后沉没，2224 名船上人员中有 1514 人死亡。

图 3-29 泰坦尼克号

灾难过后，泰坦尼克号所属的白星航运公司统计出了所有乘客的信息和生还情况。在这个案例中乘客的数量较多，而且每个乘客的信息完善，生还情况也做好了标记，可以使用机器学习来建立一个生还概率预测的模型。

2）观察数据

乘客信息数据如表 3-4 所示，只展示数据的一部分。数据中每一列的含义分别是 pclass（客舱等级）、survived（是否生还）、name（乘客姓名）、sex（性别）、age（年龄）、sibsp（同代直系亲属人数）、parch（不同代直系亲属人数）、ticket（船票编号）、fare（票价）、cabin（客舱号）、embarked（登陆港口）、boat、body、home. dest（出发地和目的地）。在这些数据中，可以思考一下，哪些可以用来做特征值，而哪些却不适合。

在接下来的实验中，选择的特征值有 survived、pclass、sex、age、sibsp、parch、ticket、fare、cabin。在这些特征中，有一些特征取值是不连续的分类数据，这类数据没有中间值量化的概念。例如 survived（是否生还）中，用 1 表示生还，用 0 表示死亡，并不存在 0.5 这种数据，中间值是没有意义的。相同的数据类型还有 sex（性别），male 是男性，female 是女性；pclass（船舱等级），1 表示一等舱，2 表示二等舱，3 表示三等舱。

确定了特征向量后，就可以基本确定这是一个多元线性模型。模型最终的训练会拟合为一个超平面。输入一个乘客的详细特征就可以预测该乘客能否在这次事故中生还。

表 3-4　泰坦尼克号乘客数据

pclass	survired	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
1	1	Allen,Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5	S	2		St Louis,MO
1	1	Allison,Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22C26	S	11		Montreal,PQ/Chesterville,ON
1	0	Allison,Miss. Helen Loraine	female	2	1	2	113781	151.5500	C22C26	S			Montreal,PQ/Chesterville,ON
1	0	Allison,Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.5500	C22C26	S		135	Montreal,PQ/Chesterville,ON
1	0	Allison,Mrs. Hudson J C(Bessie Waldo Daniels)	female	25	1	2	113781	151.5500	C22C26	S			Montreal,PQ/Chesterville,ON
1	1	Anderson,Mr. Harry	male	48	0	0	19952	26.5500	E12	S	3		New York,NY
1	1	Andrews,Miss. Komelia Theodosia	female	63	1	0	13502	77.9583	D7	S	10		Hudson,NY
1	0	Andrews,Mr. Thomas Jr	male	39	0	0	112050	0.0000	A36	S			Belfast,NI
1	1	Appleton,Mrs. Edward Dale(Charlotte Lamson)	female	53	2	0	11769	51.4792	C101	S	D		Bayside,Queens,NY
1	0	Artagaveytia,Mr. Ramon	male	71	0	0	PC17609	49.5042		C		22	Montevideo,Uruguay
1	0	Astor,Col. John Jacob	male	47	1	0	PC17757	227.5250	C62C64	C		124	New York,NY
1	1	Astor,Mrs. John Jacob(Madeleine Talmadge Force)	female	18	1	0	PC17757	227.5250	C62C64	C	4		New York,NY
1	1	Aubart,Mme. Leontine Pauline	female	24	0	0	PC17477	69.3000	B35	C	9		Paris,France
1	1	Barber,Miss. Ellen"Nellie"	female	26	0	0	19877	78.8500		S	6		
1	1	Barkworth,Mr. Algemon Henry Wilson	male	80	0	0	27042	30.0000	A23	S	B		Hessle,Yorks
1	0	Baumann,Mr. John D	male		0	0	PC17318	25.9250		S			New York,NY
1	0	Baxter,Mr. Quigg Edmond	male	24	0	1	PC17558	247.5208	B58B60	C			Montreal,PQ
1	1	Baxter,Mrs. James(Helene DeLaudeniere Chaput)	female	50	0	1	PC17558	247.5208	B58B60	C	6		Montreal,PQ
1	1	Bazzani,Miss. Albina	female	32	0	0	11813	76.2917	D15	C	8		
1	0	Beattie,Mr. Thomson	male	36	0	0	13050	75.2417	C6	C	A		Winnipeg,MN
1	1	Beckwith,Mr. Richard Leonard	male	37	1	1	11751	52.5542	D35	S	5		New York,NY
1	1	Beckwith,Mrs. Richard Leonard(Sallie Monypeny)	female	47	1	1	11751	52.5542	D35	S	5		New York,NY
1	1	Behr,Mr. Karl Howell	male	26	0	0	111369	30.0000	C148	C	5		New York,NY
1	1	Bidois,Miss. Rosalie	female	42	0	0	PC17757	227.5250		C	4		

3）将回归问题转为分类问题

对于泰坦尼克号生存预测的案例，最终的预测结果只有两个：要么为 1 生还，要么为 0 死亡。为了方便理解，用一元线性模型（票价-生存）来说明问题。如果用先前讲的回归模型来处理，拟合出来表示生死的直线一定会是一个连续的值，如图 3-30 所示。

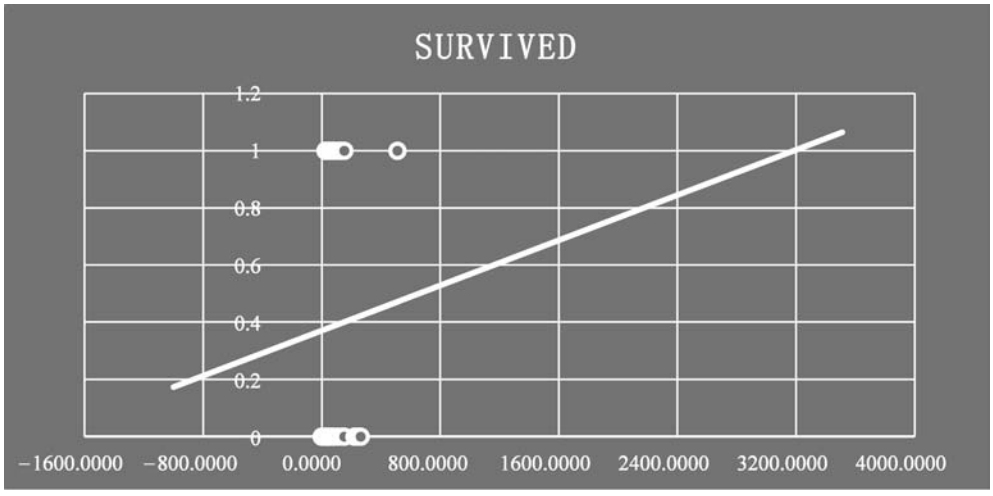


图 3-30　线性回归表示分类问题

在图 3-30 中，x 轴表示乘客的票价，y 轴表示生存与否，图中的圆点是乘客样本数据，可以看出分类数据只分布在 0 或 1 中。而拟合出来的预测曲线却超过了这个范围，尤其是当票价超过 3200 时，生存状态会大于 1，这说明把线性回归作为模型是存在问题的。

为了解决这个二分类问题，最好是能把所有的结果计算出一个[0,1]的值作为概率，当结果为 1 时即为 100%发生，当结果为 0 时表示没有发生。但仅仅这样还是不够的，想象一下，每当抛出一枚硬币，要么是正面要么是反面，但极为罕见的情况是立在桌面上的。为了符合这个实际情况，避免计算出来的结果在 0.5 附近，因为这种预测是没有意义的，提供不了参考价值，所以使结果尽可能的贴近 0 或者 1。

使用 Sigmoid 函数将线性回归线转为逻辑回归线。Sigmoid 函数如下：

$$\sigma(z)=\frac{1}{1+e^{-z}}$$

将线性模型 $y=ax1+bx2+cx3+\cdots+d$ 代入 Sigmoid 函数公式，就完成了线性回归到逻辑回归的转换。从图 3-31 可以看出，预测的概率在[0,1]，且并不容易聚集在 0.5“似是而非”这个地带。

4）模型训练与测试

具体的模型训练与搭建过程不在本书介绍，感兴趣的读者可以参照网上利用 KNIME 实现泰坦尼克号数据的预测实验。在 Python 中使用 sklearn 实现逻辑回归模型并做出预测的代码如下：

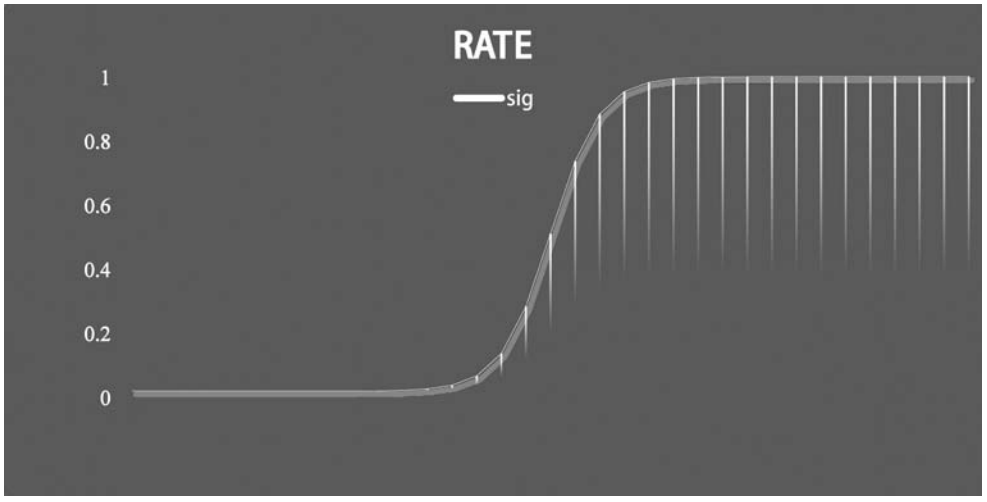


图 3-31　Sigmoid 函数

```
#基于训练集使用逻辑回归建模
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

#将模型应用于测试集并查看混淆矩阵
y_pred = classifier.predict(X_test)
#在测试集上的准确率
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(classifier.score
(X_test, y_test)))
#运行后得到精度
# Accuracy of logistic regression classifier on test set: 0.77
```

回顾电影中泰坦尼克号中的两个主人公 Jack 和 Rose，按照电影中他俩的身世给予相应的特征值如下，特征向量为[survived，name，pclass，sex，age，sibsp，parch，fare，embarked]。将这两个样本数据交给模型预测的代码如下：

```
#定义 Jack 和 Rose 的特征向量
Jack_info = [0,'Jack',3,'male',23,1,0,5.0000,'S']
Rose_info = [1,'Rose',1,'female',20,1,0,100.0000,'S']
#将 Jack 和 Rose 的特征向量加入新乘客作为预测集，并选取关键特征
new_passenger_pd = pd.DataFrame([Jack_info,Rose_info],columns = selected_cols)
#将新乘客数据集的关键特征与预测标签分离为 x_features 和 y_label
x_features,y_label = prepare_data(new_passenger_pd)
#使用关键特征 x_features 进行预测
surv_probability = model.predict(x_features)
print(surv_probability)
```

最后的输出结果为 Jack：0.118600，Rose：0.986752，可以看出运行结果和电影的结局是吻合的。

3.4 监督学习

前面介绍了线性模型进行回归和分类的原理。线性模型属于监督学习，是非常可靠的首选算法，适用于非常大的数据集，也适用于高维数据。实际上，在监督学习的模型家族中除了线性模型，还有其他各种模型，本节内容介绍监督学习中的一些经典模型。


视频讲解

3.4.1 支持向量机

在机器学习的分类问题中，有两个非常类似且平分秋色的方法：一个是前面介绍的线性逻辑回归；另一个就是支持向量机，两者在不同的应用场景有着不同的表现。对于二分类问题，线性逻辑回归和支持向量机都是通过训练具有标签的二维特征向量来生成模型，区别仅在于损失函数的实现上不同。在逻辑回归模型中，损失函数通过 Sigmoid 拟合的曲线与实际标签的差距作为衡量标准，这种损失函数称为 Logistic Loss 函数，如图 3-32 所示。

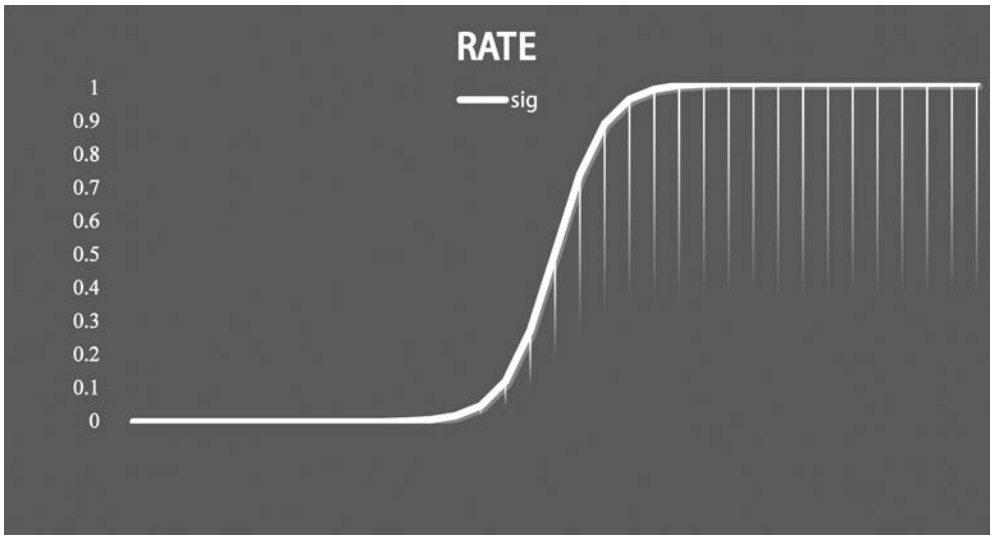


图 3-32 使用 Sigmoid 表示分类问题

在支持向量机中，损失函数通过分类支持向量之间的距离作为衡量标准，这种损失函数称为 Hinge Loss 函数。

1. 支持向量

假如有标签分为蓝球和红球几个样本数据，如图 3-33 所示，画一条直线使得将蓝球和红球进行分开，这条直线就成为支持向量。

随着样本的增加，支持向量的参数也会不断地优化调整位置，如图 3-34 所示。这一过程和线性逻辑回归的原理基本相同。

然而支持向量机相比于线性回归，对于没有明显分界面的样本数据有更好的区分能力。如图 3-35 所示，当蓝球和红球样本交叉混在一起时，使用线性回归无论怎样画出分界线都很难做到正确分类。遇到这种情况，可以将篮球在三维空间中提高一些，将红球降低一些，然后使用一个平面作为分界面来分类。对于三维空间的样本，此时支持向量就由一条直线

变为一个平面了，如图 3-36 所示。

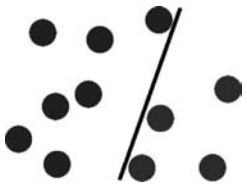


图 3-33　支持向量(见彩插)

图 3-34　支持向量调整(见彩插)

图 3-35　二维支持向量不可分的情况(见彩插)

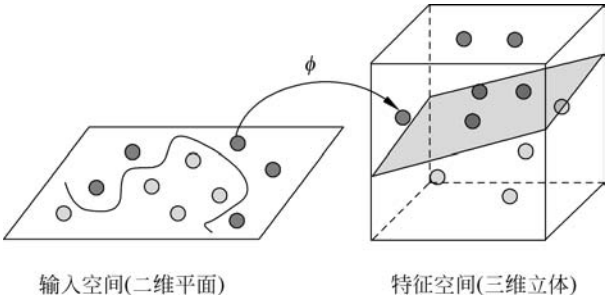


图 3-36　三维变换(见彩插)

接下来的问题就是该怎样把这些样本点根据类别变换到三维空间。这个时候就需要核函数来转换了，最常用的是利用高斯核函数，如图 3-37 所示。初学者不需要掌握核函数是如何计算的，只要明白，使用核将数据转换为另一个维度，使二维中不可线性分离的数据转换为三维空间中线性可分离的数据。就好比人们在四周道路都不通的情况下，可以通过翻墙来越过障碍物一样。

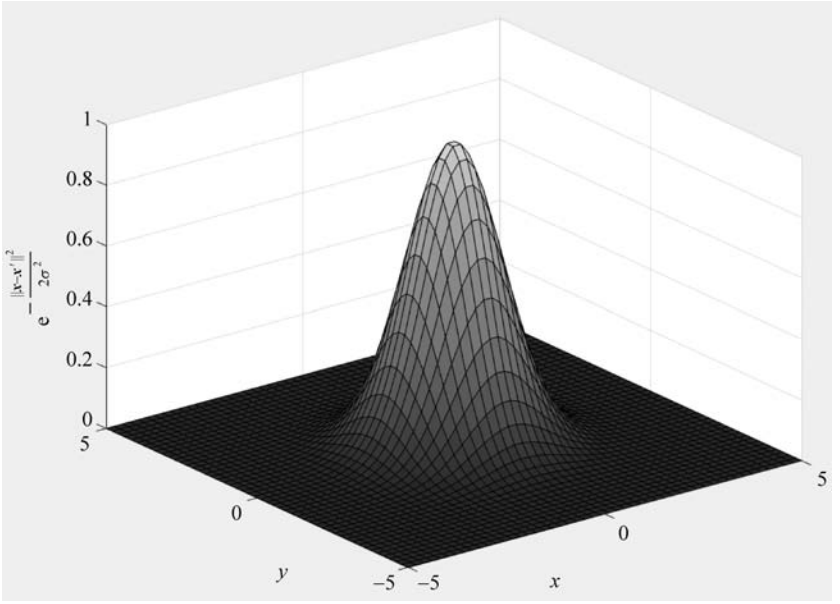


图 3-37　高斯核函数

2. 支持向量与回归线的区别

支持向量直线和线性回归拟合的直线看似相似，但由于损失函数的不同，因此优化的目的也是不同的。对于支持向量来说，要求在向量的两侧尽可能地塞下同类的样本，也可以理解为两类样本距离支持向量会尽可能远，中间的空隙尽可能大。而回归拟合的直线则是尽可能远离大多数聚集在一起的样本点。

如图 3-38 所示，蓝色样本大多数聚集在右上角，红色样本大多聚集在左下角，蓝色和红色背景色的深浅代表预测样本的概率。

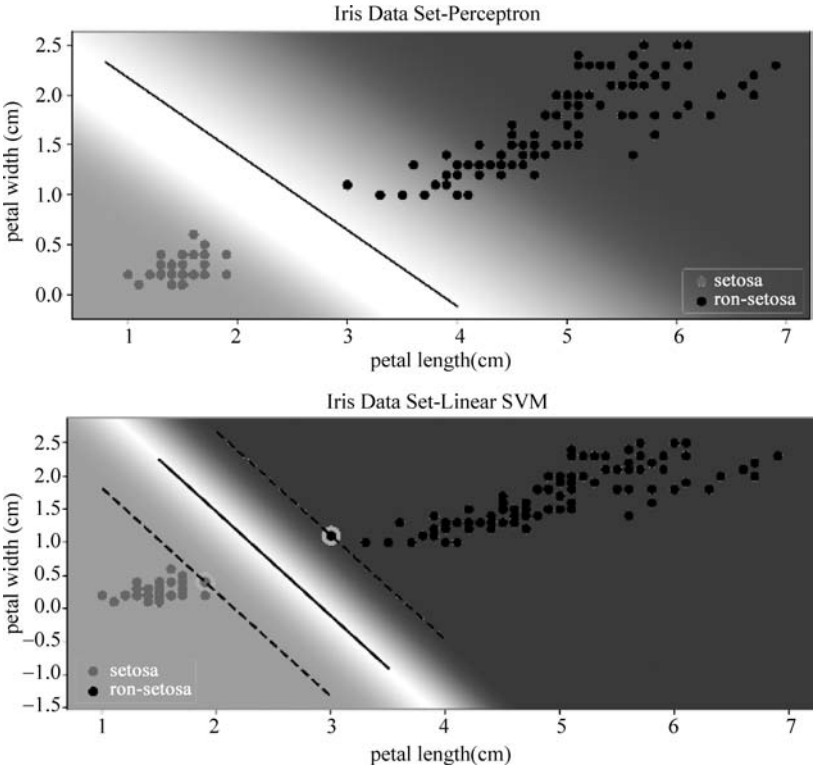


图 3-38　逻辑回归与支持向量机（见彩插）

在线性回归中，分界线尽可能远离这两个聚集区域，呈现出了向右倾斜的姿态，这代表着逻辑回归的分类更注重将特征明显的样本尽可能地正确分类。而对于中间特征模糊的样本所在区域，背景色较浅，说明线性回归对于特征模糊的样本分类效果不佳。两个样本区域颜色过渡缓慢，这是由于线性回归只告诉概率，决定权还在于人类。

在支持向量机中，分界线则是尽可能地让蓝色和红色样本之间的距离更大，对于特征模糊的样本所在区域，背景色过渡比较剧烈，这主要是由于支持向量机不能预测概率，而是绝对分类。

在实际应用中，对于小规模数据集，支持向量机的效果要好于线性回归，但是在大数据中，支持向量机的计算复杂度受到限制，而线性回归因为训练简单，使用频率更高。

3.4.2　贝叶斯分类器


视频讲解

贝叶斯分类是一类分类算法的总称，这类算法均以贝叶斯定理为基础，故统称为贝叶斯分类。而朴素贝叶斯分类是贝叶斯分类中最简单，也是常见的一种分类方法。朴素贝叶斯分类的基本原理是贝叶斯定理，通俗来说，就是计算在一个条件下发生某件事的概率。假如一个班级的两个同学到了做毕业设计的时间，两个人能力相同，独立完成毕业设计的概率都为0.8，一位同学去了企业通过实习来完成毕业设计，另一位同学则去了图书馆通过查阅书籍来完成毕业设计。对这两位同学能否完成毕业设计来分类（预测完成的概率），哪一位同学会更容易被分类为能够完成呢？很明显是去企业实习的那位同学，因为他的前提是能够有更高的概率从企业拿到毕业设计的实际案例。下面从统计学的基本概念来说明这个问题。

1. 基本术语

1）概率

概率是指用来描述某些不确定问题发生的可能性，这种可能性用0～1的数值来表示。例如抛硬币，正面朝上和背面朝上的概率是相同的，每个概率都是0.5。如果用事件A表示正面朝上，那么$P(A)$表示正面朝上的概率，即$P(A)=0.5$。

2）样本空间

样本空间表示一个事情发生的所有可能结果的集合，比如抛硬币结果的样本空间为{正面，反面}，这个集合称为全集。如果是投掷骰子，每个骰子一共有6个面，那么样本空间就是{1，2，3，4，5，6}，这个空间也是全集，如图3-39所示。

图3-39　骰子的6个面

3）条件概率

条件概率是指事件A在另外一个事件B已经发生条件下的发生概率。在掷骰子的案例中，出现4或5或6任一一面的概率为0.5，记作$P(4,5,6)=P(4)+P(5)+P(6)=1/6+1/6+1/6=0.5$。假如有人对骰子做了手脚，使得4、5、6面更容易出现，假设概率增大到$P(4,5,6)=0.8$，在这样的条件下，出现6这一面的概率明显会增大。

用A表示出现第6面的事件，用B表示骰子被做了手脚这个条件。那么在B条件下发生A的概率记作$P(A|B)$。其中的B可以理解为特征的概率，而A是分类的概率，如图3-40所示。

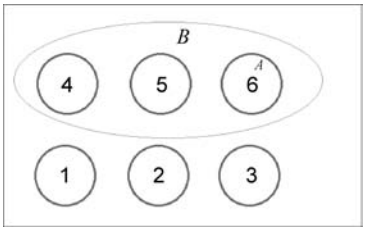


图3-40　样本空间中的条件事件

4）全概率和贝叶斯公式

如图3-41所示，全集空间划分为A_1、A_2、A_3 3个子空间时，阴影区域B的概率就为

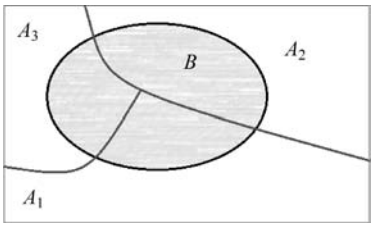


图 3-41　子空间中的条件事件

$$P(B)=P(A_1)P(B|A_1)+P(A_2)P(B|A_2)+P(A_3)P(B|A_3)$$

根据概率公式可得贝叶斯公式：

$$P(A_3 \mid B)=\frac{P(A_3B)}{P(B)}=\frac{P(B\mid A_3)P(A_3)}{P(B)}$$

$P(A_3|B)$是已知 B 发生后 A 的条件概率，也由于得到 B 的取值而被称作 A 的后验概率。$P(A_3)$是 A 的先验概率(或边缘概率)，之所以称为“先验”是因为不考虑任何 B 方面的因素。$P(B|A_3)$是已知 A 发生后 B 的条件概率，也由于得自 A 的取值而被称作 B 的后验概率。$P(B)$是 B 的先验概率或边缘概率。

2. 贝叶斯分类应用

按照图 3-41 中的案例说明概率表示为面积比例。已知一个样本空间分成 3 类，分别是 A_1、A_2、A_3，并且知道概率分别为 $P(A_1)$、$P(A_2)$、$P(A_3)$。浅灰色阴影部分表示样本中具有 B 特征的概率，在 A_1 类别中具有 B 特征的样本概率为 $P(B|A_1)$，在 A_2 类别中具有 B 特征的样本概率为 $P(B|A_2)$，在 A_3 类别中具有 B 特征的样本概率为 $P(B|A_3)$。问题就变成了，具有 B 特征的样本是 A_1、A_2、A_3 的概率是多少，利用贝叶斯公式，就可以计算出它们的概率。哪个概率大，就认为属于哪一类。

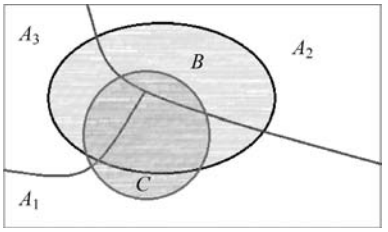


图 3-42　子空间中多个条件事件

在实际应用过程中，仅有一个 B 特征时最终的预测结果并不准确，当增加一个 C 特征时，如图 3-42 所示，得到了在 A_1、A_2、A_3 类别中具有 C 特征的样本概率分别为 $P(C|A_1)$、$P(C|A_2)$、$P(C|A_3)$，此时样本同时具有了 B 特征和 C 特征，那么它的分类概率就变成如下公式：

$$\begin{aligned}P(A_1 \mid B\text{\^{}}C)&=P(B\text{\^{}}C \mid A_1)\times P(A_1)/P(B\text{\^{}}C)\\&=P(B \mid A_1)\times P(C \mid A_1)\times P(A_1)/P(B)/P(C)\\P(A_2 \mid B\text{\^{}}C)&=P(B\text{\^{}}C \mid A_2)\times P(A_2)/P(B\text{\^{}}C)\\&=P(B \mid A_2)\times P(C \mid A_2)\times P(A_2)/P(B)/P(C)\\P(A_3 \mid B\text{\^{}}C)&=P(B\text{\^{}}C \mid A_3)\times P(A_1)/P(B\text{\^{}}C)\\&=P(B \mid A_3)\times P(C \mid A_3)\times P(A_3)/P(B)/P(C)\end{aligned}$$

可以看出，随着新特征加入，样本在整体空间的面积不断缩小，概率的计算会更加准确。贝叶斯分类器只适用于分类问题，比线性模型速度快，适用于非常大的数据集和高维数据，但精度通常低于线性模型。

视频讲解

3.4.3　决策树

1. 决策树简介

决策树是广泛用于分类和回归任务的模型，主要用于分类。决策树模型呈现树形结构，是基于输入特征对实例进行分类的模型。下面通过一个判定客户是否买车的案例来了解决

策树，如图 3-43 所示。

决策树是一棵树，每个父节点都有两个或两个以上子节点，决策树中最末端的节点称为叶节点，而叶节点就是分类结果，除了叶节点外的其他节点代表了样本的特征。在这个例子中，只要给出客户的样本信息{性别，年龄}就可以对他是否买车做出预测。本质上，在掌握了程序设计基础相关的知识点后就可以比较简单地实现决策树模型，就是从一层层的 if/else 问题中进行学习，并得出结论。

图 3-43　决策树

2. 决策树生成

在实际的应用案例中，决策树模型的使用流程一般分为 4 个步骤：生成决策树模型、产生分类规则、测试模型、预测模型。与线性模型不同的是，决策树模型的生成过程是一个从根节点到叶节点不断深入迭代生成的过程。下面通过一组数据来分析，如表 3-5 所示。

表 3-5　样本数据

用户 ID	年　龄	性　别	收入/万	婚姻状况	是否买房
1	27	男	15	否	否
2	47	女	30	是	是
3	32	男	12	否	否
4	24	男	45	否	是
5	45	男	30	是	否
6	56	男	32	是	是
7	31	男	15	否	否
8	23	女	30	是	否

由表 3-5 可以直观看出，数据的特征空间为{年龄，性别，收入，婚姻状况}，而要预测的分类是一个二分类问题。在决策树中，级别越高的节点包含的信息量越小，不确定性越大，因此分类越模糊；而级别越低的节点包含的信息量越大，不确定性越小，因此分类就会越具体。根据决策树的原理，目的就是要把信息量小的特征尽可能放到级别高的节点上，把信息量大的特征尽可能放到级别低的节点上。

1）熵

熵是一个热力学概念，用来描述事物的混乱程度，可以理解为描述一个信息是否具体。如图 3-44 所示，3 个图中鸡蛋的数量是一样的，很明显是第一幅图能够更明显的数出鸡蛋的数量，因为信息更具体，事物更有序，称为“熵最小”。而对于第三张图，信息不明显，事物更无序，称为“熵最大”。

2）类别熵与特征类别熵

案例中是否买房的情况分成了两类，且买房与不买房的人数是不同的。为了衡量样本空间中分类的分离度，引入了类别熵的概念，公式如下：

图 3-44　不同程度的熵

$$H(X)=-\left(P(x_1)\log_2\frac{1}{P(x_1)}+P(x_2)\log_2\frac{1}{P(x_2)}+\cdots+P(x_n)\log_n\frac{1}{P(x_n)}\right)$$

在买房的数据中心，一共 3 人买房，5 人没买，那么依据公式，就可以计算出买房情况的类别熵为 $H(C)=-\frac{3}{8}\log_2\frac{3}{8}-\frac{5}{8}\log_2\frac{5}{8}=0.288$。

同贝叶斯分类思想类似，类别熵必然受到特征的影响，比如引入年龄这个因素时，类别熵的情况就会发生变化，称为特征类别熵，用 $H(C|X)$表示。

对于年龄因素，可以分为 3 个区间（特征类别），分别是 x_{11}：20～30 岁，x_{12}：30～40 岁和 x_{13}：大于 40 岁，此时的数据表则变为 3 个数据表，如表 3-6～表 3-8 所示。

表 3-6　x_{11} 区间样本

用户 ID	年　龄	性　别	收入/万	婚姻状况	是否买房
1	27	男	15	否	否
4	24	男	45	否	是
8	23	女	30	是	否

表 3-7　x_{12} 区间样本

用户 ID	年　龄	性　别	收入/万	婚姻状况	是否买房
3	32	男	12	否	否
7	31	男	15	否	否

表 3-8　x_{13} 区间样本

用户 ID	年　龄	性　别	收入/万	婚姻状况	是否买房
2	47	女	30	是	是
5	45	男	30	是	否
6	56	男	32	是	是

根据类别熵的公式可以计算出三者的特征熵为

$$H(C\mid x_{11})=-\frac{1}{3}\log_2\frac{1}{3}-\frac{2}{3}\log_2\frac{2}{3}=0.278$$

$$H(C\mid x_{12})=-\frac{0}{2}\log_2\frac{0}{2}-\frac{2}{2}\log_2\frac{2}{2}=0$$

$$H(C\mid x_{13})=-\frac{2}{3}\log_2\frac{2}{3}-\frac{1}{3}\log_2\frac{1}{3}=0.278$$

样本中在30～40这个年龄区间，特征熵为0，实际意义就表示这个特征的信息混乱无序度为0，给予的信息是确定的，30～40这个年龄段没有人买房。

最终可以计算出年龄 X_1 的特征类别熵为

$$\begin{aligned}H(C\mid X_1)&=P(x_{11})H(C\mid x_{11})+P(x_{12})H(C\mid x_{12})+P(x_{13})H(C\mid x_{13})\\&=\frac{3}{8}\times 0.278+\frac{2}{8}\times 0+\frac{3}{8}\times 0.278\\&=0.209\end{aligned}$$

同理，计算出的性别 X_2、收入 X_3、婚姻状况 X_4 的特征类别熵分别为

$$H(C\mid X_2)=0.284$$
$$H(C\mid X_3)=0.151$$
$$H(C\mid X_4)=0.274$$

3）信息增益

信息增益就是在某个特征条件下，信息熵减少的程度。直观的理解是，当知道确定某个特征时，由于信息量提高，那么分类的结果会更加具体，无序程度也就降低了。它的意义表示为特征 X 对分类 C 的贡献度大小，用公式表示为

$$G(X)=H(C)-H(C\mid X)$$

通过公式计算可得年龄、性别、收入、婚姻状况4个特征的薪资增益为

$$G(X_{\text{年龄}})=0.079$$
$$G(X_{\text{性别}})=0.004$$
$$G(X_{\text{收入}})=0.137$$
$$G(X_{\text{婚姻状况}})=0.014$$

可以看到，收入这个特征的信息增益最大，所以将收入作为根节点，此时生成的分支如图3-45所示。

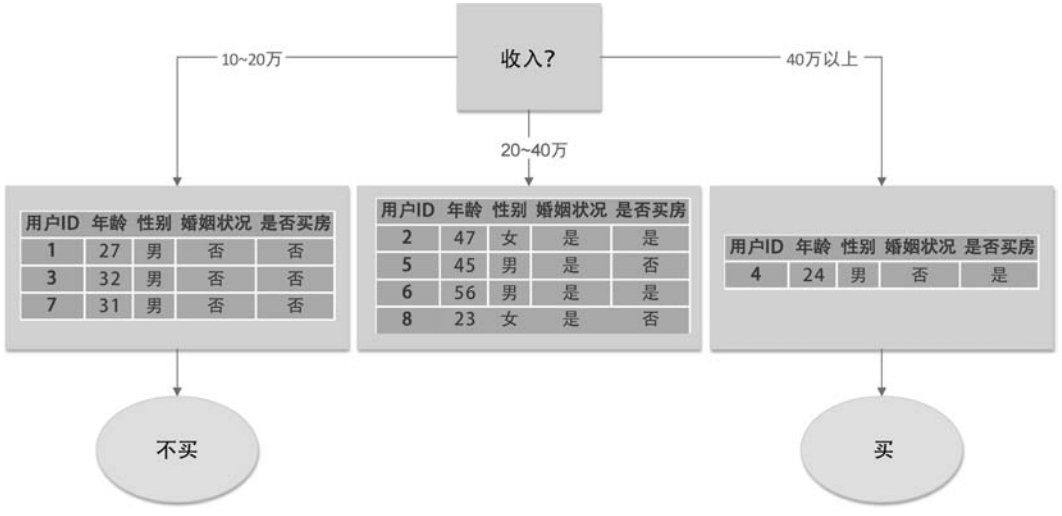


图3-45　确定根节点后的决策树

当生成新的页节点时，可以看到，在收入大于 40 万的节点中，分类只有一种情况——买，这时就可以生成最终节点“买”；在收入介于 10 万～20 万的节点中，分类也只有一种情况——不买，这时也可以生成最终节点“不买”。在 20 万～40 万这个节点，分类有多种情况，这时对于这个节点的样本再次重复决策树生成的过程就可以。

决策树有两个优点：一是得到的模型很容易可视化，非专家也很容易理解（至少对于较小的树而言）；二是算法完全不受数据缩放的影响。

决策树的主要缺点在于很容易过拟合，泛化性能很差。因此，在大多数应用中，往往使用集成方法来替代单棵决策树。


视频讲解

3.4.4 神经网络

应用于人工智能领域的神经网络一般称为人工神经网络（Artificial Neural Network，ANN）。人工神经网络的研究很早就出现了，当时是从信息处理角度通过对人脑神经元及其网络进行模拟、简化和抽象，建立某种模型，按照不同的连接方式组成不同的网络。人工神经网络是一种运算模型，由大量的节点（或称神经元）相互连接构成。神经网络研究的进展，在很大程度上并不是受益于计算机科学家的贡献，而是来源于生物学家、心理学家。随着人们对大脑研究的不断进步，人们意识到如果要让机器模拟人类的智能，突破点应该是让机器模拟人类的神经结构。可以看出，人工智能已是一个规模相当大的、多学科交叉的学科领域。

1. 神经元模型

人类的智能活动主要是靠大脑来实现的，大脑中的神经系统是由 140～160 亿个神经元细胞构成的，如图 3-46 所示。神经元中的树突用来接收其他神经元发来的信号（神经递质），这些树突可以看作神经元的输入部分。出当神经元接收到信号的刺激后，就会通过轴突末梢发出信号，这些轴突末梢可以看作神经元的输出部分，这些信号再去刺激其他的神经元。比较神奇的是，只有当神经元受到足够强度的刺激，才会响应并释放出刺激其他神经元的递质，如果受到的刺激不足，神经元将不会有输出。这样的输出其实等同于一个二分类，并不会出现中间值。上亿的神经元细胞按照不同的结构连接在一起，就形成了一个神经网络，如图 3-47 所示。

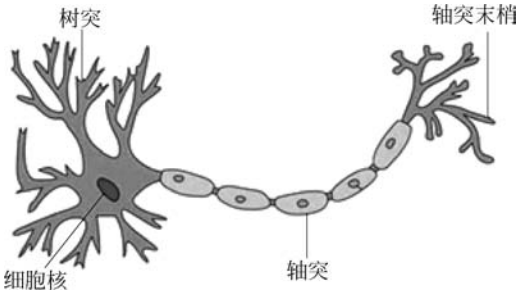


图 3-46 神经元细胞

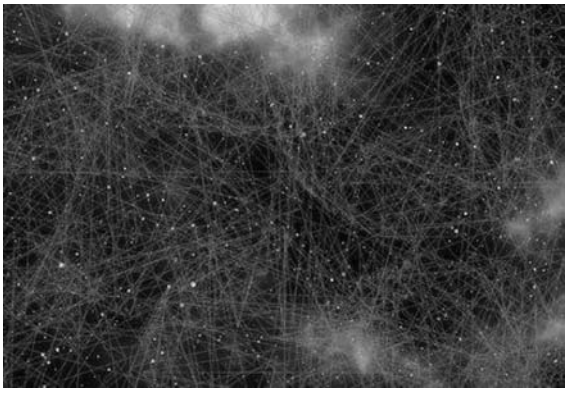


图 3-47　神经元之间的连接情况

2. 人工神经网络模型

人工神经网络是如何模拟生物神经网络的呢？回顾之前讲到的多元线性分类模型。假如样本空间为$\{x_1,x_2,x_3,x_4\}$，那么线性模型就可以表示为 $y=w_1x_1+w_2x_2+w_3x_3+w_4x_4+b$。线性模型的训练过程就是求出参数$(w_1,w_2,w_3,w_4,\theta)$。这个计算模型称为感知器，如图 3-48 所示。为了模拟神经元的二分类输出，再将 y 值进行 Sigmoid 处理，如图 3-49 所示。在这里，Sigmoid 称为激活函数。

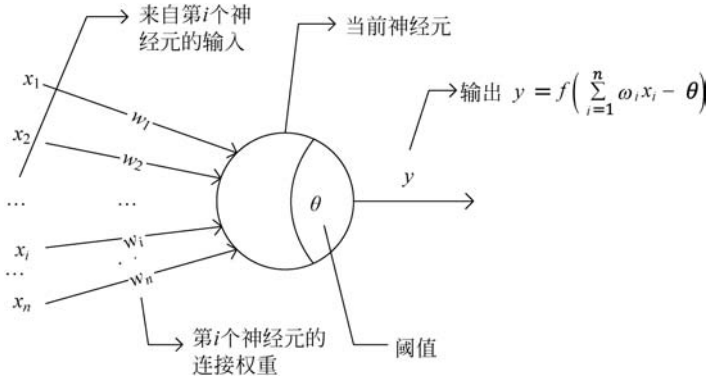


图 3-48　感知器模型

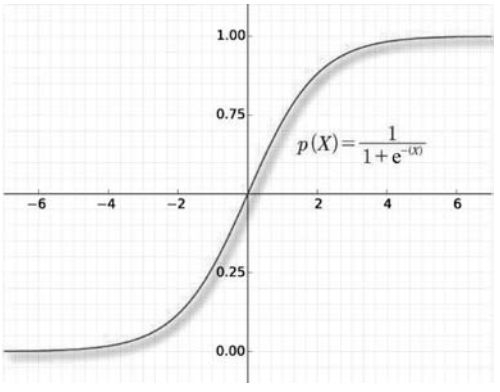


图 3-49　Sigmoid 激活函数

可以看出，上面的线性逻辑分类模型，就是最基本的神经元模型的实现，有 n 个输入，每一个输入对应一个权值 w，神经元内会对输入与权重做乘法后求和，求和的结果加上偏置 θ（阈值，减少误差），最终将结果放入激活函数中，有激活函数给出最后的输入，输出的结果往往是二进制的，0 代表一致，1 代表激活。这样的机制称为感知机。如果将这些神经元按照有层次的组织起来，就形成了神经网络，如图 3-50 所示。在图中，Layer1 代表输入的 3 个特征，可以发现每一个特征都会与后面每一个神经元做连接，这些连接都有不同的权重 w，这样的连接成为全连接。Layer2 表示 5 个神经元组成的第二层网络，与第三层全连接。在最后一层 Layer4，表示输出层，这一层的 4 个神经元往往代表不同的分类，每个神经元的输出是自己所代表分类的概率。

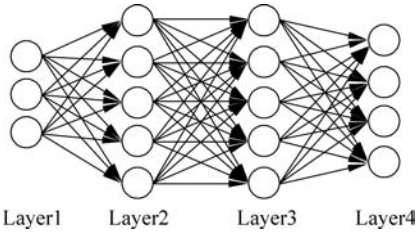


图 3-50　输入层、隐层、输出层三层网络模型

图 3-50 中的 Layer2 和 Layer3 在输入层和输出层中间，称为隐层。想象大脑的工作机制，当看图片来说出图片内容时，实际上是眼睛视网膜的神经元作为输入层受到刺激，这些信号在大脑中穿过多层的神经网络后，刺激才传到嘴巴上，从而说出了结果，隐层的作用就在于此。早期的神经网络，由于计算机性能较弱，计算效率低，对于一个三层网络训练就会花费大量的时间和成本。随着现在计算机性能的提高，增加多个隐层渐渐变得可行（对，模型也是算法）这种有更多隐层神经网络的模型称为深度学习。

3. 神经网络种类

由于神经元的有多个输入和一个输出，可以像拼积木一样以非常高的自由度组建各种形状的模型。上面介绍的网络模型是一个标准的完全连接神经网络。在实际应用中，不同的组建方式在不同的案例上性能会有不同的适应性。根据连接方式的不同，可以将神经网络分为卷积神经网络（Convolutional Neural Networks，CNN）、循环神经网络（Recurrent Neural Network，RNN）和深度信念网络（Deep Belief Network，DBN）。

1）卷积神经网络

当用神经网络来进行图片识别时，输入的特征向量就是图片中的每一个像素。用手机拍到的照片，不可以直接输入模型。对于一个只有 32 像素的图片，建立的三层模型如图 3-51 所示，一条连接便代表一个需要计算的权重值。现在的手机拍照像素都很高，一张 2000 万像素的照片就意味着要把 2000 万个特征输入模型，那么所需要计算的权重值就会成为一个天文数字，这是不能接受的。

针对这种情况，要实现两个目的：第一，尽可能地缩小原始图片的尺寸；第二，在缩小原始尺寸图片的时候要保证尽可能地保留图片中的特征细节。这时，就需要用到卷积。

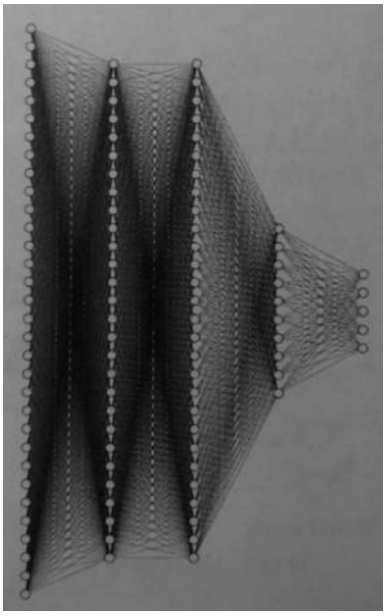


图 3-51　32×32 图像的全连接网络模型

卷积的原理可以理解为某一确定条件下的值，是由其他条件共同决定的。举一个例子，某同学在前天向喜欢的女生表白找到了女朋友，这件事使他的开心度为 x_1；这位同学在昨天得知自己考试成绩是第一名，这件事使他的开心度为 x_2；在今天，他又知道自己在技能大赛中获得了第一名，这件事使他的开心度为 x_3，试问这位同学在今天到底有多开心？是 x_3 吗，当然不是，因为前两天的开心“余韵”还在，只是程度降低了。所以他在今天的开心度可以表示为 $x_3+w_2x_2+w_1x_1$，这样的计算过程就是卷积。

在图 3-52 中把下方的 7×7 大小的图片缩小为 3×3 的图片，并不是简单地去掉多余的像素，而是每一个新像素都是由邻域的像素共同计算得出的。使用卷积缩放图片，在改变图片大小后，图片清晰度没有发生变化。

由此可知，卷积的值是由于邻域的值按照不同的权重共同计算而得出来的。那么应该计算哪些邻域，权重又该是多少，这就是卷积核的概念。不同的卷积核不仅决定了图片缩放的大小，还可以提取不同的图片特征。在图 3-53 中，图 3-53(b)为应用低通滤波器，会看到图片变得更模糊；图 3-53(c)表示应用卷积核为高通滤波器，图片的细节得到加强；图 3-53(d)表示使用的卷积核为边缘检测，丢掉了图像中的色彩信息，加强了轮廓信息。可以看出，对于边缘检测卷积核，即使图片缩小后，这些特征信息也非常明显，因此常用来作为神经网络的输入特征。

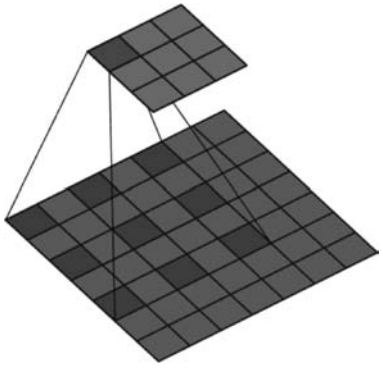


图 3-52　卷积原理

(a) Lena原图　(b) 低通滤波器　(c) 高通滤波器　(d) 边缘检测

图 3-53　不同卷积核的效果

LeNet-5 是 Yann LeCun 在 1998 年设计的用于手写数字识别的卷积神经网络。当年美国大多数银行使用它来识别支票上面的手写数字，它是早期卷积神经网络中最具代表性的模型之一。

LeNet-5 共有 7 层（不包括输入层），每层都包含不同数量的训练参数，如图 3-54 所示。

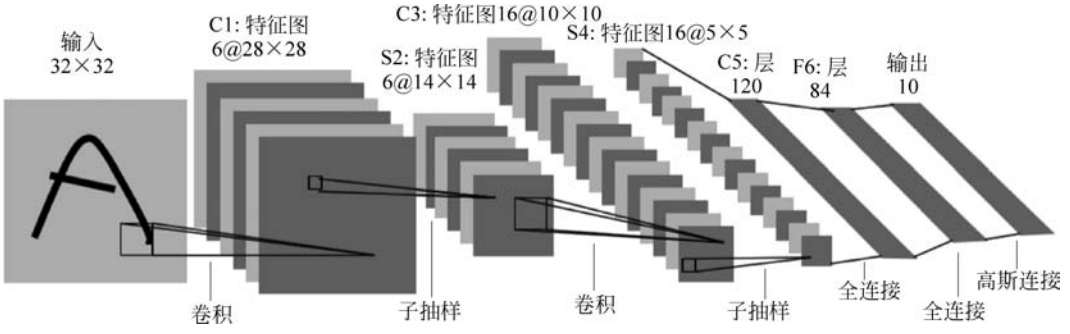


图 3-54　LeNet-5 模型

然而，由于当时缺乏大规模的训练数据，计算机的计算能力也跟不上，而且网络结构相对过于简单，LeNet-5 对于复杂问题的处理结果并不理想。

CNN 在机器学习、语音识别、文档分析、语言检测和图像识别等领域有着广泛的应用。其中，CNN 在图像处理和图像识别领域取得了很大的成功，在国际标准的 ImageNet 数据集上，许多成功的模型都是基于 CNN 的。CNN 相较于传统的图像处理算法的好处之一在于：避免了对图像复杂的前期预处理过程，可以直接输入原始图像。

2）循环神经网络

有了像卷积网络这样表现非常出色的网络，为什么还需要其他类型的网络呢？传统的神经网络对于很多问题难以处理。比如用卷积神经网络做一个人脸识别的宿舍门禁系统，可以很好地识别出某个同学来决定是否放行。假如有校外人员，想拿着某同学的照片骗过门禁系统，该如何防止呢？这里就存在一个序列顺序问题。一般情况下，同学们都结伴而行，比如 a、b、c、d 4 名同学经常一起回宿舍，而有一天校外人员拿着同学 e 的照片，按照 a、b、c、e、d 的顺序混入宿舍，就不符合正常的序列顺序。卷积神经网络并没有考虑到这个序列问题，而循环神经网络就能很好地解决这个问题。之所以称为循环神经网路，即一个序列当前的输出与前面的输出也有关。再比如要预测句子的下一个单词是什么，一般需要用到前面的单词，因为一个句子中前后单词并不是独立的。具体的表现形式为网络会对前面的

信息进行记忆并应用于当前输出的计算中，即隐层之间的节点不再是无连接而是有连接的，并且隐层的输入不仅包括输入层的输出还包括上一时刻隐层的输出。理论上，RNN 能够对任何长度的序列数据进行处理。如图 3-55 所示，这是一个简单的 RNN 的结构，可以看到隐层自己是可以跟自己进行连接的。

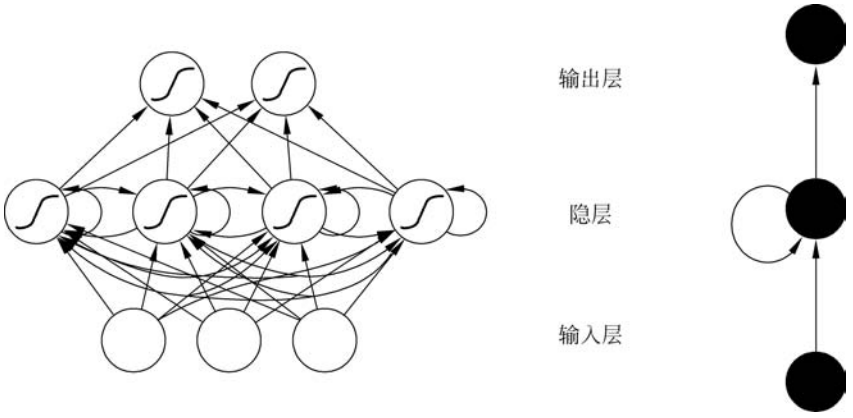


图 3-55　循环网络模型

RNN 多用于序列前后有关联的领域：

- 语言建模和文本生成——给出一个词语序列，试着预测下一个词语的可能性。这在翻译任务中是很有用的，因为最有可能的句子将是可能性最高的单词组成的句子。
- 机器翻译——将文本内容从一种语言翻译成其他语言使用了一种或几种形式的 RNN。所有日常使用的实用系统都用了某种高级版本的 RNN。
- 语音识别——基于输入的声波预测语音片段，从而确定词语。
- 生成图像描述——RNN 一个非常广泛的应用是理解图像中发生了什么，从而做出合理的描述。这是 CNN 和 RNN 相结合的作用。CNN 做图像分割，RNN 用分割后的数据重建描述。这种应用虽然基本，但可能性是无穷的。
- 视频标记——可以通过一帧一帧地标记视频进行视频搜索。

3）深度信念网络

深度信念网络是一个概率生成模型，与传统的判别模型的神经网络相对，生成模型是建立一个观察数据和标签之间的联合分布，对 P（Observation | Label）和 P（Label | Observation）都做了评估，而判别模型仅仅评估了后者，也就是 P(Label|Observation)。

深度信念网络(DBN)通过采用逐层训练的方式，解决了深层次神经网络的优化问题，通过逐层训练为整个网络赋予了较好的初始权值，使得网络只要经过微调就可以取得最优解。模型如图 3-56 所示，DBN 主要有多个受限玻尔兹曼机构成，在这里只做了解即可。

深度信念网络诞生于 2006 年，是为了解决当时神经网络的性能问题而提出的新方法，相对于传统的神经网络，它的特点主要有训练时间短，在样本数据量少时预测更精确。

随着近几年计算机硬件设备性能的提高，同时也出现了很多适合于神经网络运算的专用处理器，神经网络的发展进入了一个繁荣时期。很多优秀模型不断被提出，它们往往不是单一的网络类型，更多的是不同网络模型以不同的维度组合，这种灵活的创新方式创造了机器学习领域许多更加实用的案例。如图 3-56 所示。

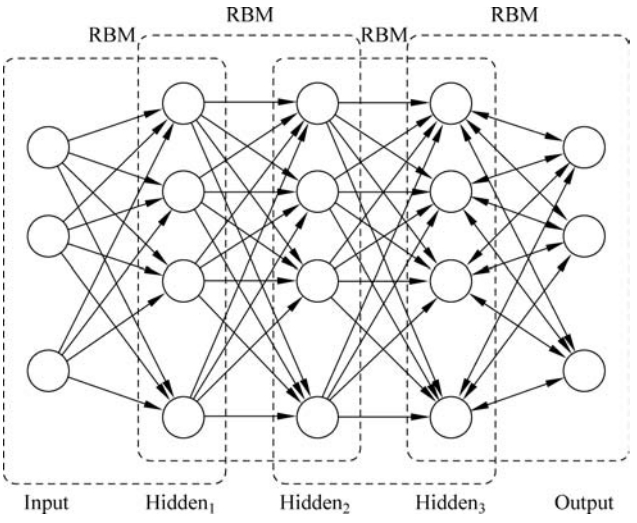


图 3-56　DNB 网络模型

3.5　机器学习案例——猫狗分类

本节内容使用 TensorFlow 和 Keras 建立一个猫狗图片分类器，如图 3-57 所示。

图 3-57　猫狗图片

3.5.1　安装 TensorFlow 和 Keras 库

TensorFlow 是一个采用数据流图（data flow graphs），用于数值计算的开源软件库。节点（Node）在图中表示数学操作，图中的线（edges）则表示在节点间相互联系的多维数据数组，即张量（tensor）。这种灵活的架构让你可以在多种平台上展开计算，例如，台式计算机中的一个或多个 CPU（或 GPU）、服务器、移动设备等。TensorFlow 最初由 Google 大脑

小组(隶属于 Google 机器智能研究机构)的研究员和工程师们开发出来,用于机器学习和深度神经网络方面的研究,但这个系统的通用性使其也可广泛用于其他计算领域,如图 3-58 所示。

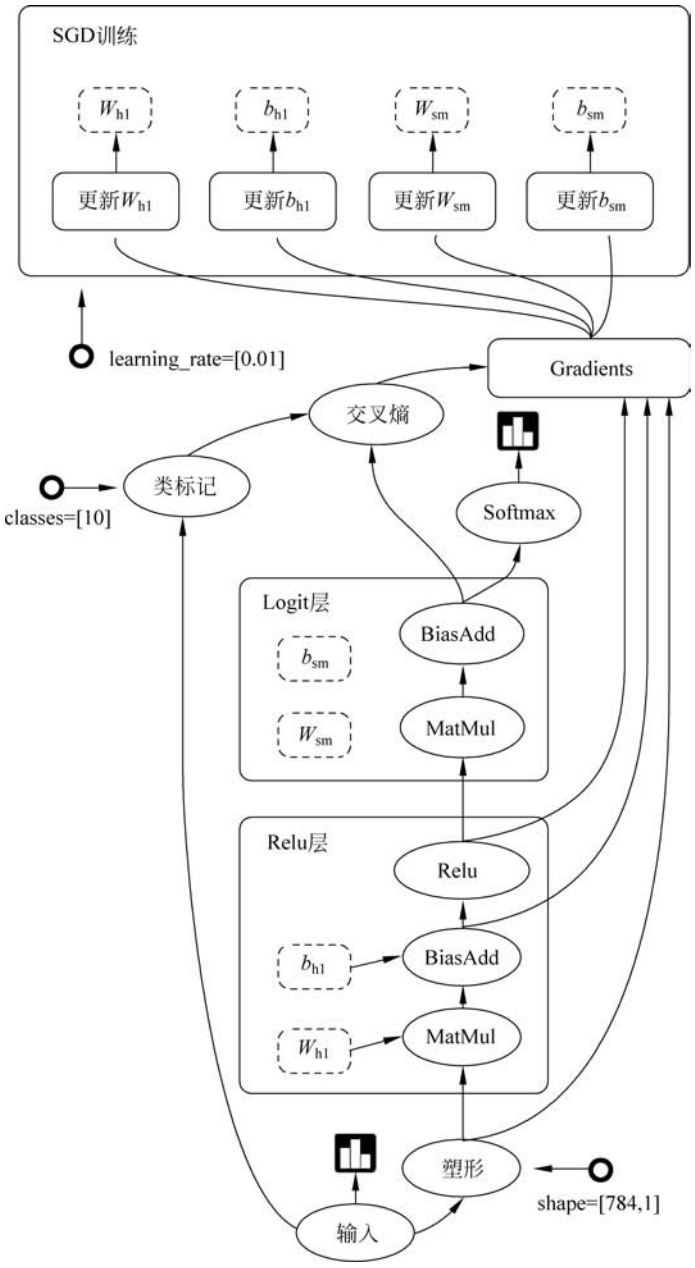


图 3-58　TensorFlow 数据流图

运行 Anaconda Prompt 工具,在该工具命令窗口下安装数值计算开源库(tensorflow),如图 3-59 和图 3-60 所示,输入以下 pip 命令:

```
pip install tensorflow
```

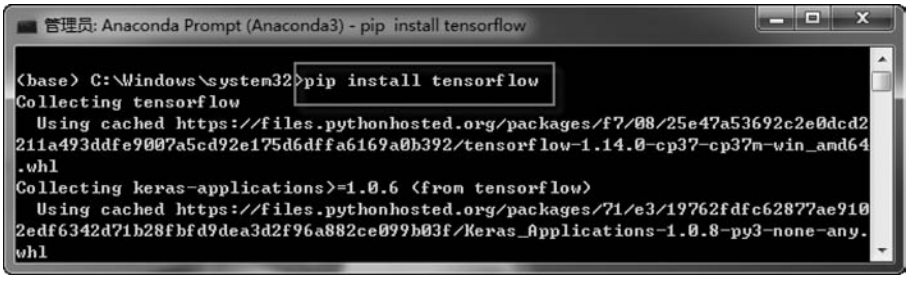


图 3-59 安装 TensorFlow 库

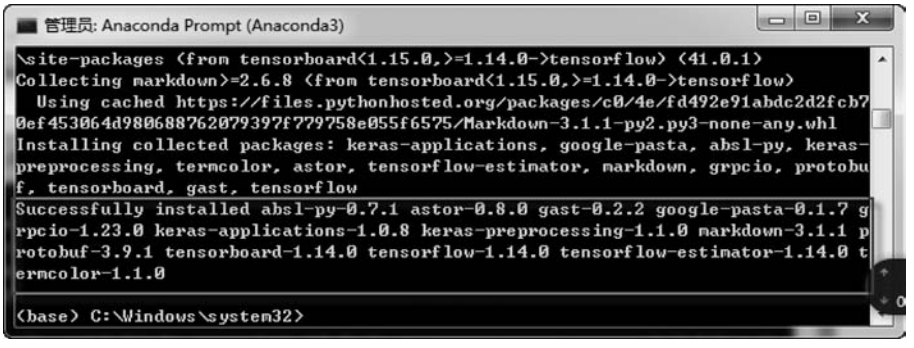


图 3-60 TensorFlow 库安装成功

Keras 是一个由 Python 编写的开源人工神经网络库，可以作为 TensorFlow、Microsoft-CNTK 和 Theano 的高阶应用程序接口，进行深度学习模型的设计、调试、评估、应用和可视化，即高层神经网络 API。

Keras 的命名来自古希腊语“κέρας（牛角）”或“κραίνω（实现）”，意为将梦境化为现实的“牛角之门”。Keras 的最初版本以 Theano 为后台，设计理念参考了 Torch 但完全由 Python 编写，自 2017 年起，Keras 得到了 TensorFlow 团队的支持，其大部分组件被整合至 TensorFlow 的 Python API 中。在 2018 年 TensorFlow 2.0.0 公开后，Keras 被正式确立为 TensorFlow 高阶 API，即 tf.keras。此外自 2017 年 7 月开始，Keras 也得到了 CNTK 2.0 的后台支持。

Keras 在代码结构上由面向对象方法编写，完全模块化并具有可扩展性，其运行机制和说明文档有将用户体验和使用难度纳入考虑，并试图简化复杂算法的实现难度。Keras 支持现代人工智能领域的主流算法，包括前馈结构和递归结构的神经网络，也可以通过封装参与构建统计学习模型。在硬件和开发环境方面，Keras 支持多操作系统下的多 GPU 并行计算，可以根据后台设置转化为 TensorFlow、Microsoft-CNTK 等系统下的组件。Keras 为支持快速实验而生，能够把你的设想迅速转换为结果。

运行 Anaconda Prompt 工具，在该工具命令窗口下安装开源人工神经网络库（keras），输入以下 pip 命令：

```
pip installkeras
```

ketas 库的安装如图 3-61 和图 3-62 所示。

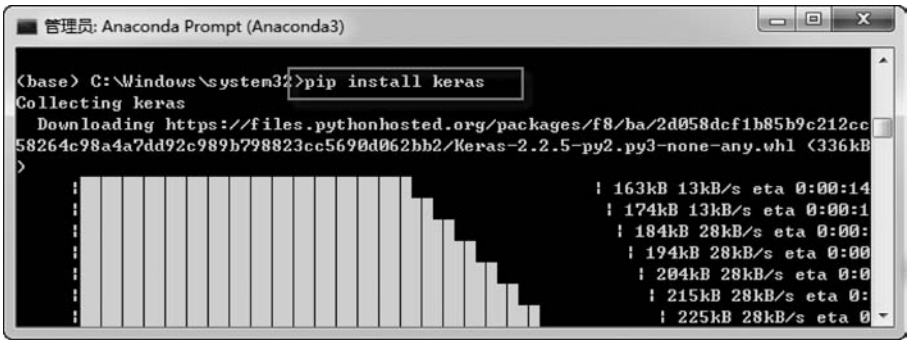


图 3-61　安装 keras 库

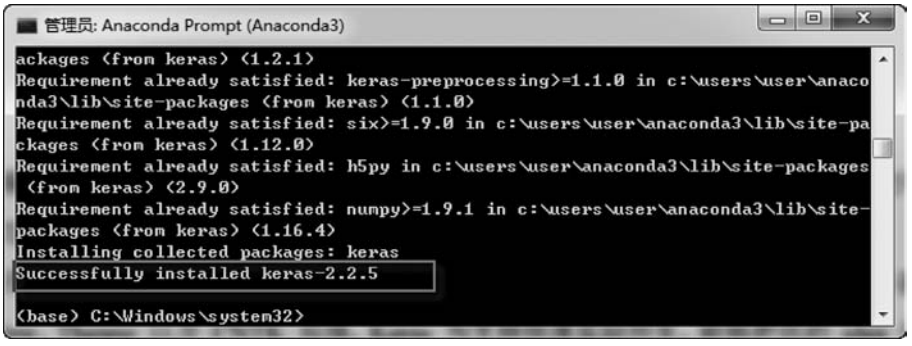


图 3-62　keras 库安装成功

环境配置好后，开始着手建立一个可以将猫狗图片分类的卷积神经网络，并使用到深度学习框架 TensorFlow 和 Keras，配置网络以便远程访问 Jupyter Notebook。

3.5.2　案例实现

【目的】

- 面向小数据集，使用 Keras 进行数据增强。
- 使用 Keras 构建两层卷积神经网络，实现猫狗分类。
- 对学习模型进行性能评估。

【原理】

基于 Keras 框架，构建顺序神经网络，包含三层卷积，使用 ReLU 激活函数，同时使用 max 池化，后接 2 个全连接层，中间使用 Dropout 进行降采样。

【训练时长】

50 个 epoch，每个 epoch 大概需要 50s，在 it1080 下训练过程共需要 40min 左右。

【数据资源】

cat_dog 数据集，从 kaggle 猫狗数据集中随机抽取 3000 个样本，其中训练集包含猫狗

各1000，测试集包含猫狗各500。

【步骤1】 数据预处理

使用小数据集（几百张到几千张图片）构造高效、实用的图像分类器的方法，需要通过一系列随机变换进行数据增强，可以抑制过拟合，使得模型的泛化能力更好。

在Keras中，这个步骤可以通过keras.preprocessing.image.ImageGenerator来实现：

（1）在训练过程中，设置要施行的随机变换。

（2）通过.flow或.flow_from_directory(directory)方法实例化一个针对图像batch的生成器，这些生成器可以被用作Keras模型相关方法的输入，如fit_generator、evaluate_generator和predict_generator。

导入相关资源的代码如下。

```
import numpy as np
from keras.preprocessing.image import ImageDataGenerator #图像预处理
import keras.backend as K
K.set_image_dim_ordering('tf')
from keras.models import Sequential
from keras.layers import Convolution2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
加载数据的代码如下:
使用.flow_from_directory()来从jpgs图片中直接产生数据和标签
#用于生成训练数据的对象
train_gen = ImageDataGenerator(
        rescale = 1./255,
        shear_range = 0.2,
        zoom_range = 0.2,
        horizontal_flip = True)

#用于生成测试数据的对象
test_datagen = ImageDataGenerator(rescale = 1./255)
```

参数解释如下：

- rotation_range是一个0～180的度数，用来指定随机选择图片的角度。
- width_shift和height_shift用来指定水平和竖直方向随机移动的程度，这是两个0～1的比例。
- rescale值将在执行其他处理前乘到整个图像上，图像在RGB通道都是0～255的整数，这样的操作可能使图像的值过高或过低，所以将这个值定为0～1的数。
- shear_range是用来进行剪切变换的程度。
- zoom_range用来进行随机放大。
- horizontal_flip随机地对图片进行水平翻转，这个参数适用于水平翻转不影响图片语义的时候。
- fill_mode用来指定当需要进行像素填充，如旋转、水平和竖直位移时，如何填充新出现的像素。

```
#从指定路径小批量读取数据
train_data = train_gen.flow_from_directory(
        '../datas/min_data/train',                    # t路径
        target_size=(150, 150),                       #图片大小
        batch_size=32,                                #每次读取的数量
        class_mode='binary')                          #标签编码风格

#从测试集中分批读取验证数据
val_data= test_gen.flow_from_directory(
        '../datas/min_data/test',
        target_size=(150, 150),
        batch_size=32,
        class_mode='binary')
```

【步骤 2】 构建神经网络

模型包含三层卷积加上 ReLU 激活函数，再接 max-pooling 层，代码如下。

```
model = Sequential()
model.add(Convolution2D(32, 3, 3, input_shape=(3, 150, 150)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

两个全连接网络，并以单个神经元和 Sigmoid 激活结束模型。这种选择会产生二分类的结果，与这种配置相适应，使用 binary_crossentropy 作为损失函数，每个卷积层的滤波器数目并不多。为了防止过拟合，需要添加 Dropout 层，代码如下。

```
model.add(Flatten())                #展平
model.add(Dense(64))                #全连接
model.add(Activation('relu'))       #激活
model.add(Dropout(0.5))             #降采样
model.add(Dense(1))                 #全连接
model.add(Activation('sigmoid'))    #激活

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

投入数据，训练网络，代码如下。

```
model.fit_generator(
        train_data,                          #训练集
        samples_per_epoch = 2000,            #训练样本数量
        nb_epoch = 50,                       #训练次数
        validation_data = val_data,          #验证数据
        nb_val_samples = 800                 #验证集大小
    )
model.save_weights('../model/cat_dog.h5')  #保存模型文件
```

完整代码如下。

【案例 3-1】 Conv_3.ipynb

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K


#图片大小
img_width, img_height = 150, 150

train_data_dir = '../datas/min_data/train'
validation_data_dir = '../datas/min_data/test'
nb_train_samples = 2000
nb_val_samples = 1000
epochs = 50
batch_size = 16

if K.image_data_format() == 'channels_first':
    input_shape = (3, img_width, img_height)
else:
    input_shape = (img_width, img_height, 3)

#数据读取、数据增强
train_gen = ImageDataGenerator(
    rescale = 1. / 255,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True)
val_gen = ImageDataGenerator(rescale = 1. / 255)

train_data = train_gen.flow_from_directory(
    train_data_dir,
    target_size = (img_width, img_height),
    batch_size = batch_size,
    class_mode = 'binary')

val_data = val_gen.flow_from_directory(
    validation_data_dir,
    target_size = (img_width, img_height),
    batch_size = batch_size,
```

```
    class_mode = 'binary')

#构建三层卷积神经网络
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape = input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

#添加站平层、全连接层、降采样层
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

#定义损失函数和优化策略
model.compile(loss = 'binary_crossentropy',
              optimizer = 'rmsprop',
              metrics = ['accuracy'])

#开始训练
model.fit_generator(
    train_data,
    steps_per_epoch = nb_train_samples // batch_size,
    epochs = epochs,
    validation_data = val_data,
    validation_steps = nb_val_samples // batch_size)
#保存
yaml_string = model.to_yaml()
with open('cat_dog.yaml', 'w') as outfile:
    outfile.write(yaml_string)
model.save_weights('cat_dog.h5')
```

运行结果如下：

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
WARNING:tensorflow:From /home/yangrh/anaconda3/lib/python3.6/site-packages/keras/backend/
tensorflow_backend.py:1208: calling reduce_prod (from tensorflow.python.ops.math_ops) with
keep_dims is deprecated and will be removed in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
```

```
WARNING:tensorflow:From /home/yangrh/anaconda3/lib/python3.6/site-packages/keras/backend/
tensorflow_backend.py:1297: calling reduce_mean (from tensorflow.python.ops.math_ops) with
keep_dims is deprecated and will be removed in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
Epoch 1/50
125/125 [==============================] - 40s - loss: 0.7057 - acc: 0.5140 -
val_loss: 0.6797 - val_acc: 0.5363
Epoch 2/50
125/125 [==============================] - 40s - loss: 0.6851 - acc: 0.5865 -
val_loss: 0.6391 - val_acc: 0.6839
…
Epoch 50/50
125/125 [==============================] - 40s - loss: 0.3996 - acc: 0.8370 -
val_loss: 0.5988 - val_acc: 0.7947
```

【性能评估】

50 个 epoch，每个 epoch 大概需要 50s，在 it1080 下训练过程共需要 40min 左右，在验证集上的准确率达到 80%，训练准确率与错误率如图 3-63 所示。

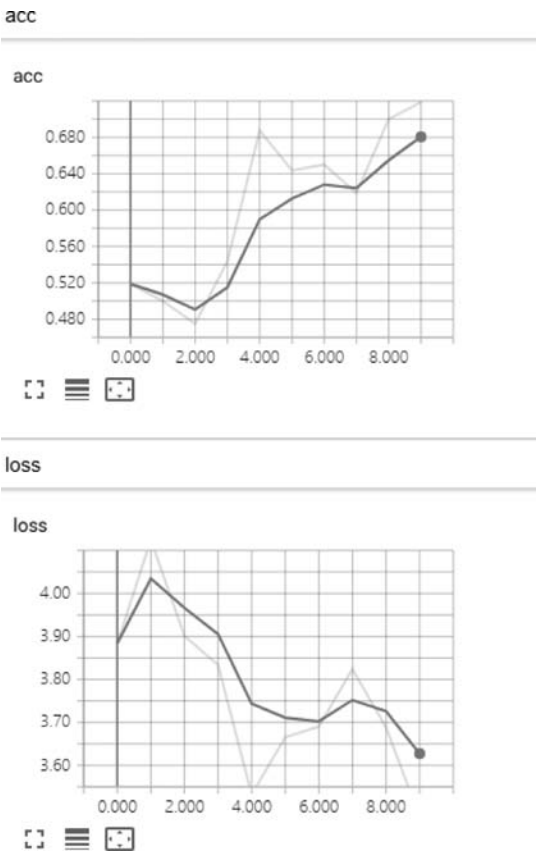


图 3-63　准确率与错误率

下面案例代码使用训练好的模型对猫狗图片进行分类。

【案例 3-2】 Predict_Model. ipynb

```
import os
import keras
import keras.backend as K
from tables.tests.test_tables import LargeRowSize
K.set_image_dim_ordering('tf')
import numpy as np
from keras import callbacks
from keras.models import Sequential, model_from_yaml, load_model
from keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPool2D
from keras.optimizers import Adam, SGD
from keras.preprocessing import image
from keras.utils import np_utils, plot_model
from sklearn.model_selection import train_test_split
from keras.applications.resnet50 import preprocess_input, decode_predictions

img_h, img_w = 150, 150
image_size = (150, 150)
nbatch_size = 256
nepochs = 48
nb_classes = 2

def pred_data():

    with open('cat_dog.yaml') as yamlfile:
        loaded_model_yaml = yamlfile.read()
    model = model_from_yaml(loaded_model_yaml)
    model.load_weights('cat_dog.h5')

    sgd = Adam(lr=0.0003)
    model.compile(loss='categorical_crossentropy',optimizer=sgd, metrics=['accuracy'])

    images = []
    path='../datas/pred/'
    for f in os.listdir(path):
        img = image.load_img(path + f, target_size=image_size)
        img_array = image.img_to_array(img)

        x = np.expand_dims(img_array, axis=0)
        x = preprocess_input(x)
        result = model.predict_classes(x,verbose=0)

        print(f,result[0])

# 0表示猫,1表示狗
pred_data()
```

运行结果如下：

```
c7.jpg [0]
dog.33.jpg [0]
dog.34.jpg [1]
dog.41.jpg [0]
c3.jpg [0]
c1.jpg [1]
dog.36.jpg [0]
c4.jpg [0]
dog.35.jpg [0]
dog.39.jpg [0]
c5.jpg [0]
dog.38.jpg [0]
c2.jpg [0]
dog.40.jpg [0]
c6.jpg [1]
c0.jpg [0]
dog.37.jpg [0]
```

本章总结

- 机器学习(Machine Learning,ML)是人工智能的核心分支。
- 机器学习的核心就是数据、算法(模型)和算力(计算机计算能力)。
- 赫布学习理论是神经网络(Neural Network,NN)形成记忆痕迹的首要基础,也是最简单的神经元(Neuron)学习规则。
- 从功能上来说,机器学习的功能大致可分为回归、分类和聚类。
- 机器学习的学习方式大体分为 3 类：监督学习、无监督学习和弱监督学习。
- 机器学习的流程包括分析案例、数据获取、模型训练和模型验证这 4 个过程。
- 线性模型(Linear Model)蕴含了机器学习中一些重要的基本思想,线性模型形式简单、易于建模,具有很好的解释性。
- 特征不明显、小规模数据集,采用支持向量机的效果要好于线性回归。
- 贝叶斯分类以贝叶斯定理为基础。
- 决策树模型呈现树形结构,是基于输入特征对实例进行分类的模型。
- 人工神经网络(Artificial Neural Network,ANN)是从信息处理角度通过对人脑神经元及其网络进行模拟、简化和抽象,建立某种模型,按照不同的连接方式组成不同的网络。
- 根据连接方式的不同,可以将神经网络分为卷积神经网络、循环神经网络和深度信念网络。

本章习题

1. 机器学习的简称是(　　)。

A. AI　　B. ML　　C. DL　　D. NN

2. 机器学习的核心要素包括(　　)。

A. 数据　　B. 操作人员　　C. 算法　　D. 算力

3. 按照学习方式的不同,可以将机器学习分为以下哪类?(　　)

A. 监督学习　　B. 无监督学习　　C. 弱监督学习　　D. 聚类

4. 决策树中的分类结果是最末端的节点,这些节点称为(　　)。

A. 根节点　　B. 父节点　　C. 子节点　　D. 叶节点

5. 机器学习的流程包括分析案例、数据获取、(　　)和模型验证这4个过程。

A. 数据清洗　　B. 数据分析　　C. 模型训练　　D. 模型搭建

6. 梯度为(　　)的点,就是损失函数的最小值点,一般认为此时模型达到了收敛。

A. −1　　B. 0　　C. 1　　D. 无穷大

7. 使用下面哪个函数可以将线性回归线转为逻辑回归线?(　　)

A. Sigmoid　　B. 高斯核函数　　C. P(A)　　D. H(x)

8. 支持向量机的简称是(　　)。

A. AI　　B. ML　　C. ANN　　D. SVM

9. 下面不属于人工神经网络的是(　　)。

A. 卷积神经网络　　B. 循环神经网络

C. 网络森林　　D. 深度信念网络

10. 当数据特征不明显,数据量少的时候,采用下面哪个模型?(　　)

A. 线性回归　　B. 逻辑回归　　C. 支持向量机　　D. 神经网络