普通高等教育新工科人才培养规划教材

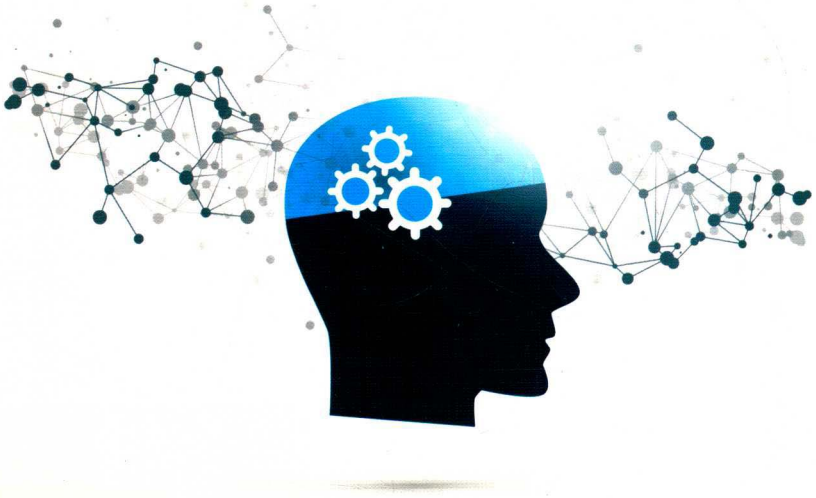
（大数据专业）

深度学习

——卷积神经网络

算法原理与应用

王改华 ◆ 编著



中国水利水电出版社
www.waterpub.com.cn

普通高等教育新工科人才培养规划教材（大数据专业）

深度学习——卷积神经网络算法原理与应用

王改华　编著

中国水利水电出版社
www.waterpub.com.cn
·北京·

内 容 提 要

考虑到近几年深度学习的快速发展，而此方面的教材缺乏，本书以卷积神经网络算法原理为基础，对最近几年提出的卷积神经网络进行系统介绍。

本书较全面地介绍了卷积神经网络的基本内容，注重卷积神经网络的基本概念、基本原理和网络结构的阐述。全书共分为九章，第 1 章～第 3 章介绍了深度学习及卷积神经网络的概念及发展，卷积神经网络相关的数学基础知识，神经网络的基础算法原理等知识点；第 4 章、第 5 章对卷积神经网络的基本原理及扩展机制进行剖析；第 6 章介绍了自编码器的一些基本原理及算法；第 7 章针对卷积神经网络的优化算法进行了详细的分析说明；第 8 章、第 9 章是卷积神经网络的典型结构及卷积神经网络的压缩算法应用。

附录中增加了部分典型卷积神经网络结构的 Matlab 及 Python 程序，结合实际、突出应用，旨在帮助使用者加深对基本概念的理解和提高综合问题分析的能力。

全书内容丰富，层次分明，主要面向人工智能及相关专业的高年级本科生及研究生，也可做为从事深度学习的软件工程师的参考书目。

图书在版编目（CIP）数据

深度学习 : 卷积神经网络算法原理与应用 / 王改华编著. -- 北京 : 中国水利水电出版社, 2019.4
普通高等教育新工科人才培养规划教材. 大数据专业
ISBN 978-7-5170-7595-0

Ⅰ. ①深… Ⅱ. ①王… Ⅲ. ①机器学习－高等学校－教材 Ⅳ. ①TP181

中国版本图书馆CIP数据核字(2019)第069204号

策划编辑：周益丹　责任编辑：张玉玲　加工编辑：高双春　封面设计：梁　燕

书　　名	普通高等教育新工科人才培养规划教材（大数据专业） **深度学习——卷积神经网络算法原理与应用** SHENDU XUEXI——JUANJI SHENJING WANGLUO SUANFA YUANLI YU YINGYONG
作　　者	王改华　编著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	三河市铭浩彩色印装有限公司
规　　格	184mm×260mm　16 开本　9.75 印张　170 千字
版　　次	2019 年 4 月第 1 版　2019 年 4 月第 1 次印刷
印　　数	0001—3000 册
定　　价	29.00 元

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

前　言

深度学习是机器学习研究中的一个新的领域，其动机在于建立、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。自2006年深度学习概念被提出以来，其以极高的发展速度在各行各业得到了广泛的应用。基于深度学习的新技术、新算法不断涌现。本书以深度学习中卷积神经网络算法原理为基础，对最近几年提出的卷积神经网络进行系统介绍。

本书力求体系完整、通俗易懂，书中系统地讲述了卷积神经网络的基本原理，从前向传播、反向传播、优化算法等方面详细阐述。同时对卷积神经网络和自编码器的常用算法进行介绍。针对算法实例应用进行分析。

本书资料搜集工作：袁国亮搜集整理第6章，吕朦搜集整理第4章、第8章，刘文洲搜集整理第5章，郑旭、万溪洲、郭钊搜集整理第2章、第3章、第7章、第8章。

本书在内容上注重精选、结合实际、突出应用。主要面向人工智能及相关专业的高年级本科生及研究生，也可做为从事深度学习的软件工程师的参考书目。

由于编者水平有限，加之时间仓促，书中难免存在不当和谬误之处，恳请有关专家和广大读者不吝赐教。

编　者

2019年1月

目　　录

第 1 章　绪论

1.1　深度学习

1.1.1　概述

深度学习是机器学习中一种基于对数据进行表征学习的方法。观测值（例如一幅图像）可使用多种方式来表示，如每个像素强度值的向量，或者更抽象地表示成一系列边、特定形状的区域等。含多隐层的多层感知器就是一种深度学习结构，通过组合低层特征形成更加抽象的高层表示属性，以发现数据的分布式特征表示。使用某些特定的表示方法更容易从实例中学习任务，例如，人脸识别或面部表情识别等，它的最终目标是让机器能够像人一样具有分析学习能力，能够识别文字、图像和声音等数据。

深度学习经历了三次发展浪潮：20 世纪 40 年代到 60 年代深度学习的雏形出现在控制论中，20 世纪 80 年代到 90 年代深度学习表现为联结主义，直到 2006 年深度学习的概念由 Hinton 等人正式提出。

深度学习（Deep Learning）是利用多层神经网络结构，从大数据中学习现实世界中各类事物，能直接用于计算机计算的表示形式，被认为是智能机器的“大脑结构”。

Learning 是让计算机自动调整参数以拟合函数的过程；Deep 是多个函数进行嵌套，构成一个多层神经网络，利用非监督贪心逐层训练算法有效地自动调整函数参数。简单地说深度学习就是使用多层神经网络进行机器学习，动机在于建立、模拟人脑进行分析学习、解释数据的过程，例如图像、声音和文本的解译。

同机器学习方法一样，深度学习也有监督学习与无监督学习之分，不同的学习框架下建立的学习模型不同。例如，卷积神经网络（Convolutional Neural Networks，CNNs）就是一种深度监督学习下的机器学习模型；深度置信网（Deep Belief Nets，DBNs）是一种无监督学习下的机器学习模型。

1.1.2 基本思想

深度学习的前身是从神经科学的角度出发的简单线性模型。这些模型被设计为使用一组 n 个输入 $x_1,...,x_n$ 并将它们与一个输出 y 相关联。模型希望学习一组权重 $w_1,...,w_n$，并计算它们的输出 $f(x,w)=x_1\times w_1+...+x_n\times w_n$。线性模型有很多局限性，最著名的是，它们无法学习异或(XOR)函数，即 $f([0,1],w)=1$，$f([1,0],w)=1$，$f([1,1],w)=0$ 和 $f([0,0],w)=0$。

20 世纪 80 年代初期，大多数认知科学家研究符号推理模型。尽管这很流行，但符号模型很难解释大脑如何真正使用神经元实现推理功能。联结主义者开始研究基于神经系统实现的认知模型，其中很多复苏的想法可以追溯到心理学家 Donald Hebb 在 20 世纪 40 年代的工作。

联结主义的几个关键概念在今天的深度学习中仍然是非常重要的。其中一个概念是分布式表示。系统的每一个输入都应该由多个特征表示，并且每一个特征都应该参与到多个可能输入的表示。另一个重要成就是反向传播在训练具有内部表示的深度神经网络中的成功使用。

2006 年，Geoffrey Hinton 表明深度信念网络可以使用一种称为贪婪逐层预训练的策略来有效地训练。假设我们有一个系统 S，它有 n 层 $(S_1,...,S_n)$，它的输入是 I，输出是 O，形象地表示为：$I=>S_1=>S_2=>...=>S_n=>O$，如果输出 O 等于输入 I，即输入 I 经过这个系统变化之后没有任何的信息损失。

设处理 a 信息得到 b，再对 b 处理得到 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。

保持不变，这意味着输入 I 经过每一层 S_i 都没有任何的信息损失，即在任何一层 S_i，它都是原有信息（即输入 I）的另外一种表示。通过调整系统中参数，使得它的输出仍然是输入 I，那么就可以自动地获取到输入 I 的一系列层次特征，即 $(S_1,...,S_n)$。对于深度学习来说，其思想就是堆叠多个层，也就是说这一层的输出作为下一层的输入。通过这种方式，可以实现对输入信息进行分级表达。另外，假设输出严格地等于输入，这个限制太严格，可以略微地放松这个限制，例如只要使得输入与输出的差别尽可能地小即可。

1.1.3 基本分类

深度学习算法主要分类有监督学习算法和无监督学习算法。

1. 监督学习

判断是否是监督学习，就看输入数据是否有标签。利用一些训练数据，使机器能够总结出一些规律，然后用这些规律来分析未知数据。设输入为 x，输出为 y，我们期望的输出为 y^*，可定义实际输出与期望输出的差别来评价神经网络的好坏。例如将两个向量距离的平方定义为这个差别

$$d(y^*, y) = \frac{1}{2} | y^* - y |^2 \tag{1.1}$$

由此定义损失函数为

$$loss(w) = \sum_{i=1}^{N} d(y^*, y) \tag{1.2}$$

$$\hat{w} = \arg\min loss(w) \tag{1.3}$$

在数学上，这是一个无约束优化问题。如果用测试集上的损失函数来调整参数，会出现严重过拟合。所以通常准备两个集合，一个是测试集，一个是训练集。参数的学习只针对训练集，找到使训练集损失尽量小的参数，然后在测试集上测试该组参数面对训练集之外的样本的表现。

使用监督学习方法对浅层网络进行训练通常能够使参数收敛到合理的范围内。有监督训练需要依赖于有标签的数据才能进行训练。然而有标签的数据通常是稀缺的，因此对于许多问题，很难获得足够多的样本来拟合一个复杂模型的参数。例如，考虑到深度网络具有强大的表达能力，在不充足的数据上进行训练将会导致过拟合。

2. 无监督学习

现实生活中常常会有这样的问题：缺乏足够的先验知识，因此难以人工标注类别或进行人工类别标注的成本太高。很自然地，我们希望计算机能代我们完成这些工作，或至少提供一些帮助。根据类别未知的训练样本解决模式识别中的各种问题，称为无监督学习。

目前深度学习中的无监督学习主要分为两类，一类是确定型的自编码方法及其改进算法，其目标主要是能够从抽象后的数据中尽量无损地恢复原有数据；一类是概率型的受限波尔兹曼机及其改进算法，其目标主要是使受限玻尔兹曼机达到稳定状态时原数据出现的概率最大。

1.2 卷积神经网络技术的发展与应用

1.2.1 卷积神经网络的发展

Lecun 等人提出的 LeNet-5 模型作为卷积神经网络中第一个最为熟知的经典卷积神经网络，此模型是现在卷积神经网络的原型。2012 年，Krizhevsky 等提出 AlexNet 模型，从网络规模上大大超越了 LeNet-5。另外，AlexNet 选择了大型图像分类数据库 ImageNet 作为训练数据集；在去过拟合方面，AlexNet 引入了 dropout，一定程度上减轻了网络过拟合问题。Simonyan 等在 AlexNet 的基础上，提出了 VGG 网络，通过对比不同深度网络在图像分类应用中的性能，证明了网络深度的提升有助于提高图像分类的准确度。然而，在恰当的网络深度基础上继续增加网络的层数，会带来训练误差增大的网络退化问题。针对深度网络的退化问题，He 等提出了一种 ResNet 网络结构。ResNet 通过 short connections 将低层的特征图 x 直接映射到高层的网络中。Szegedy 等更关注通过优化网络结构从而降低网络的复杂程度。他们提出了一种卷积神经网络的基本模块，称为 Inception。使用 Inception 模块构建的 GoogLeNet 的训练参数数量只有 AlexNet 的 1/12，但在 ImageNet 上的图像分类准确度却高出 AlexNet 大约 10%。此外，Springenberg 等对卷积神经网络下采样层存在的必要性提出了质疑，并设计了不含下采样层的“完全卷积网络”。“完全卷积网络”在结构上相比于传统的卷积神经网络结构更加简单，但是其网络性能却不低于带有下采样层的传统模型。2016 年，谷歌研发的基于深度神经网络和搜索树的智能机器人“AlphaGo”在围棋上击败了韩国围棋九段李世石。2017 年，Hinton 提出了 CapsNet 网络结构在 MNIST 数据集上取得高达 99.75%的识别率。

纵观卷积神经网络的整个发展历程，尽管其发展时间较短，但是卷积神经网络不论从理论还是实际应用上都得到了极大的发展。从理论发展上，我们看到其主要是网络结构与训练方法上的改进，通过扩展网络规模来提取更加深层次的图像特征，从而能更好地完成图像处理；在训练方法上，主要是对有监督学习算法以及无监督学习算法进行改进或相互结合，从而提升网络训练效率。

1.2.2 卷积神经网络的应用

卷积神经网络不仅具有传统神经网络较好的容错性、较强的自学习能力和自适应性等优点，还具有自动提取特征、权值共享以及输入图像与网络结构结合良好等优势。在手写字体识别、人脸识别、物体识别、医疗诊断、机器人、自动驾驶等视觉分析任务中都取得了比拟人类认知的结果。

1.3 自编码器的发展及其应用

1.3.1 自编码器的发展

1986 年 Rumelhart 提出自编码器的概念，并用于高维复杂数据处理，促进了神经网络的发展。2006 年，Hinton 对原型自动编码器结构进行改进，进而产生了深度学习自编码器，利用无监督逐层贪婪训练算法完成对隐含层的预训练，然后用 BP 算法对整个神经网络进行系统参数优化调整，显著降低了神经网络的性能指数，有效改善了 BP 算法易陷入局部最小的不良状况。2007 年，Bengio 提出稀疏自动编码器的概念，进一步深化自动编码器的研究。2008 年，Vincent 提出降噪自动编码器，在输入数据中添加腐坏向量，防止出现过拟合现象，并取得良好的效果。2009 年，Bengio 总结已有的深度结构，阐述利用堆叠自动编码器构建深度学习神经网络的一般方法。2010 年，Salah 对升维和降维的过程加以限制，提出收缩自动编码器。2011 年，Jonathan 提出卷积自动编码器，用于构建卷积神经网络。2012 年，Taylor 对自动编码器与无监督特征学习之间的联系进行深入的探讨，详细介绍了如何利用自动编码器构建不同类型深度结构。Hinton、Bengio 和 Vincent 等人对比了原型自动编码器、稀疏自动编码器、降噪自动编码器、收缩自动编码器、卷积自动编码器和 RBM 等结构的性能，为以后的实践和科研提供了参考。2013 年 Tdmo 研究用不同代价函数训练的 DAE 的性能，为代价函数优化策略的发展指明了方向。

1.3.2 自编码器的应用

目前，自编码器的应用主要有两个方面，一是数据去噪，二是为进行可视化而降维。设

置合适的维度和稀疏约束，自编码器可以学习到比 PCA 等技术更有意思的数据投影。

自编码器能从数据样本中进行无监督学习，这意味着可将这个算法应用到某个数据集中，取得良好的性能，且不需要任何新的特征工程，只需要适当地训练数据。但是，自编码器在图像压缩方面表现得不好。由于在某个给定数据集上训练自编码器，因此它在处理与训练集相类似的数据时可达到合理的压缩结果，但是在压缩差异较大的其他图像时效果不佳。训练自编码器，可以使输入通过编码器和解码器后，保留尽可能多的信息，但也可以训练自编码器来使新表征具有多种不同的属性。

第 2 章　相关数学基础知识

线性代数作为数学的一个分支，广泛用于科学和工程中，掌握好线性代数对于理解和从事深度学习算法相关工作很有必要。因此，本章首先探讨一些必备的线性代数知识。

2.1　矩阵

2.1.1　基本概念

标量（scalar）：一个标量就是一个单独的数，它不同于线性代数中研究的其他大部分对象（通常是多个数的数组）。我们用斜体表示标量，且赋予小写的变量名称，当我们介绍标量时，会明确它们是哪种类型的数。比如，在定义实数标量时，我们可能会说“令 $s \in \mathbb{R}$ 表示一条线的斜率”；在定义自然数标量时，我们可能会说“令 $n \in \mathrm{N}$ 表示元素的数目”。

向量（vector）：一个向量是一列数。这些数是有序排列的。通过次序中的索引，我们可以确定每个单独的数。通常我们赋予向量粗体的小写变量名称，比如 $\boldsymbol{x}$。向量中的元素可以通过带脚标的斜体表示。向量 $\boldsymbol{x}$ 的第一个元素是 x_1，第二个元素是 x_2，等等。我们也会注明存储在向量中的元素是什么类型的。如果每个元素都属于 $\mathbb{R}$，并且该向量有 n 个元素，那么该向量属于实数集 $\mathbb{R}$ 的 n 次笛卡尔乘积构成的集合，记为 $\mathbb{R}^n$。当需要明确表示向量中的元素时，我们会将元素排列成一个方括号包围的纵列

$$x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \tag{2.1}$$

我们可以把向量看作空间中的点，每个元素是不同坐标轴上的坐标。有时我们需要索引向量中的一些元素。在这种情况下，我们定义一个包含这些元素索引的集合，然后将该集合

写在脚标处。比如，指定 x_1、x_3 和 x_6，我们定义集合 $S=\{1,3,6\}$，然后写作 x_s。我们用符号 $-$ 表示集合补集中的索引。比如 x_{-1} 表示 $\boldsymbol{x}$ 中除 x_1 外的所有元素，x_{-s} 表示 $\boldsymbol{x}$ 中除 x_1、x_3、x_6 外所有元素构成的向量。

矩阵（matrix）：矩阵是一个二维数组，其中的每一个元素被两个索引（而非一个）所确定。我们通常会赋予矩阵粗体的大写变量名称，比如 $\boldsymbol{A}$。如果一个实数矩阵高度为 m，宽度为 n，那么我们说 $A\in\mathbb{R}^{m*n}$。我们在表示矩阵中的元素时，通常以不加粗的斜体形式使用其名称，索引用逗号间隔。比如，$A_{1,1}$ 表示 $\boldsymbol{A}$ 左上的元素，$A_{m,n}$ 表示 $\boldsymbol{A}$ 右下的元素。我们通过用“:”表示水平坐标，以表示垂直坐标 i 中的所有元素。比如，$A_{i,:}$ 表示 $\boldsymbol{A}$ 中垂直坐标 i 上的一横排元素。这也被称为 $\boldsymbol{A}$ 的第 i **行**（row）。同样地，$A_{:,i}$ 表示 $\boldsymbol{A}$ 的第 i **列**（column）。当我们需要明确表示矩阵中的元素时，我们将它们写在用方括号括起来的数组中。

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \tag{2.2}$$

有时我们需要矩阵值表达式的索引，而不是单个元素。在这种情况下，我们在表达式后面接下标，但不必将矩阵的变量名称小写化。比如，$f(A)_{i,j}$ 表示函数 f 作用在 $\boldsymbol{A}$ 上输出矩阵的第 i 行第 j 列元素。

张量（tensor）：在某些情况下，我们会讨论坐标超过两维的数组。一般地，一个数组中的元素分布在若干维坐标的规则网格中，我们称之为张量。我们使用字体 **A** 来表示张量“A”。张量 **A** 中坐标为（i,j,k）的元素记作 $A_{i,j,k}$。

2.1.2 矩阵运算

1. 矩阵与向量运算

转置（transpose）：转置是矩阵的重要操作之一。矩阵的转置是以对角线为轴的镜像，这条从左上角到右下角的对角线被称为**主对角线**（main diagonal）。我们将矩阵 $\boldsymbol{A}$ 的转置表示为 A^T，定义如下：

$$(A^T)_{i,j}=A_{j,i} \tag{2.3}$$

向量可以看作只有一列的矩阵。对应地，向量的转置可以看作是只有一行的矩阵。有时，我们通过将向量元素作为行矩阵写在文本行中，然后使用转置操作将其变为标准的列向量，来定义一个向量，比如

$$x=\left[x_1,x_2,x_3\right]^T \tag{2.4}$$

标量可以看作是只有一个元素的矩阵。因此，标量的转置等于它本身，$a=a^T$。矩阵的转置可以看成以主对角线为轴的一个镜像，如式（2.5）。

$$A=\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow A^T=\begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix} \tag{2.5}$$

只要矩阵的形状一样，我们可以把两个矩阵相加。两个矩阵相加是指对应位置的元素相加，比如$C=A+B$，其中$C_{i,j}=A_{i,j}+B_{i,j}$。标量和矩阵相乘，或是和矩阵相加时，我们只需将其与矩阵的每个元素相乘或相加，比如$D=a*B+c$，其中$D_{i,j}=a*B_{i,j}+c$。

在深度学习中，我们也使用一些不那么常规的符号。我们允许矩阵和向量相加，产生另一个矩阵：$C=A+b$，其中$C_{i,j}=A_{i,j}+b_j$。换言之，向量 ***b*** 和矩阵 ***A*** 的每一行相加。这个简写方法使我们无需在加法操作前定义一个将向量 ***b*** 复制到每一行而生成的矩阵。这种隐式地复制向量 ***b*** 到很多位置的方式，被称为**广播**（broadcasting）。

2. 矩阵相乘

矩阵乘法是矩阵运算中最重要的操作之一。两个矩阵 ***A*** 和 ***B*** 的矩阵乘积（matrix product）是第三个矩阵 ***C***。为了使乘法定义良好，矩阵 ***A*** 的列数必须和矩阵 ***B*** 的行数相等。如果矩阵 ***A*** 的形状是$m\times n$，矩阵 ***B*** 的形状是$n\times p$，那么矩阵 ***C*** 的形状是$m\times p$。我们可以通过将两个或多个矩阵并列放置以书写矩阵乘法，例如

$$C=A\times B \tag{2.6}$$

具体地，该乘法操作定义为

$$C_{i,j}=\sum_k A_{i,k}B_{k,j} \tag{2.7}$$

需要注意的是，两个矩阵的标准乘积不是指两个矩阵中对应元素的乘积。不过，那样的矩阵操作确实是存在的，被称为元素对应乘积（element-wise product）或者 Hadamard 乘积（Hadamard product），记为$A\odot B$。

两个相同维数的向量 ***x*** 和 ***y*** 的**点积**（dot product）可看作是矩阵乘积x^Ty。我们可以把矩阵乘积$C=A\times B$中计算$C_{i,j}$的步骤看作是 ***A*** 的第 i 行和 ***B*** 的第 j 列之间的点积。矩阵乘积运算有许多有用的性质，从而使矩阵的数学分析更加方便。比如，矩阵乘积服从分配律：

$$A(B+C)=AB+AC \tag{2.8}$$

矩阵乘积也服从结合律：

$$A(BC)=(AB)C \tag{2.9}$$

不同于标量乘积，矩阵乘积并不满足交换律（$AB=BA$的情况并非总是满足）。然而，两个向量的**点积**（dot product）满足交换律：

$$x^T y=y^T x \tag{2.10}$$

矩阵乘积的转置有着简单的形式：

$$(AB)^T=B^T A^T \tag{2.11}$$

利用两个向量点积的结果是标量，标量转置是自身的事实，我们通过式（2.12）可以证明式（2.10）

$$x^T y=(x^T y)^T=y^T x \tag{2.12}$$

由于本书的重点不是线性代数，我们并不试图展示矩阵乘积的所有重要性质，但读者应该知道矩阵乘积还有很多有用的性质。现在我们已经知道了足够多的线性代数符号，可以表达下列线性方程组：

$$Ax=b \tag{2.13}$$

其中$A\in\mathbb{R}^{m*n}$是一个已知矩阵，$b\in\mathbb{R}^m$是一个已知向量，$x\in\mathbb{R}^n$是一个我们要求解的未知向量。向量x的每一个元素x_i都是未知的。矩阵A的每一行和b中对应的元素构成一个约束。我们可以把式（2.13）重写为

$$\begin{aligned} A_{1,:}x&=b_1\\ A_{2,:}x&=b_2\\ A_{m,:}x&=b_m \end{aligned} \tag{2.14}$$

或者，更明确地，写作

$$\begin{aligned} A_{1,1}x_1+A_{1,2}x_2+\ldots+A_{1,n}x_n&=b_1\\ A_{2,1}x_1+A_{2,2}x_2+\ldots+A_{2,n}x_n&=b_2\\ A_{m,1}x_1+A_{m,2}x_2+\ldots+A_{m,n}x_n&=b_m \end{aligned} \tag{2.15}$$

矩阵向量乘积符号为这种形式的方程提供了更紧凑的表示。

3. 单位矩阵和逆矩阵

线性代数提供了被称为**矩阵逆**（matrix inversion）的强大工具。对于大多数矩阵$\boldsymbol{A}$，我们都能通过矩阵逆解析地求解式（2.13）。

为了描述矩阵逆，我们首先需要定义**单位矩阵**（identity matrix）的概念。任意向量和单位矩阵相乘，都不会改变。我们将保持n维向量不变的单位矩阵记作I_n。形式上，$I_n\in\mathbb{R}^{n*n}$，

$$\forall x\in\mathbb{R}^n, I_n x=x \tag{2.16}$$

单位矩阵的结构很简单：所有沿主对角线的元素都是1，而所有其他位置的元素都是0，

如图 2.1 所示。

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

图 2.1 单位矩阵的一个样例 I_3

矩阵 A 的矩阵逆（matrix inversion）记作 A^{-1}，其定义的矩阵满足如下条件

$$A^{-1}A=I_n \tag{2.17}$$

现在我们可以通过以下步骤求解式（2.13）：

$$Ax=b$$
$$A^{-1}Ax=A^{-1}b$$
$$I_n x=A^{-1}b$$
$$x=A^{-1}b$$

当然，这取决于我们能否找到一个逆矩阵 A^{-1}。当逆矩阵 A^{-1} 存在时，有几种不同的算法都能找到它的闭解形式。理论上，相同的逆矩阵可用于多次求解不同向量 $\boldsymbol{b}$ 的方程。然而，逆矩阵 A^{-1} 主要是作为理论工具使用的，并不会在大多数软件应用程序中实际使用。这是因为逆矩阵 A^{-1} 在数字计算机上只能表现出有限的精度，有效使用向量 $\boldsymbol{b}$ 的算法通常可以得到更精确的 $\boldsymbol{x}$。

2.2 范数

2.2.1 范数的定义

有时我们需要衡量一个向量的大小。在机器学习中，我们经常使用被称为**范数**（norm）的函数衡量向量大小。形式上，L^p 范数定义如下：

$$\|x\|_p=\left(\sum_i |x_i|^p\right)^{\frac{1}{p}} \tag{2.18}$$

其中 $p\in\mathbb{R}, p\geqslant 1$。

2.2.2 范数的分类及性质

范数（包括 L^p 范数）是将向量映射到非负值的函数。直观上来说，向量 $\boldsymbol{x}$ 的范数衡量从