

**本科毕业设计（论文）外文翻译译文**

**学生姓名**： 王荣超

**院 （系）：** 计算机学院

**专业班级**： 软件1903班

**指导教师**： 肖红

**完成日期**： 2023 年 3月 7日

**深度学习：实践者的方法**

**Deep Learning: A Practitioner's Approach**

**作者： J*osh* P*atterson***

**起止页码：1-34页**

**出版日期（期刊号）：**2017年7月27日

**出版单位：**O’R*eilly* M*edia*, I*nc*

外文翻译译文：

第 1 章 机器学习回顾

# 学习的机器

过去十年，人们对机器学习的兴趣呈爆炸式增长。几乎每天都会在计算机科学课程、行业会议以及《华尔街日报》中遇到“机器学习”这个词。在所有关于机器学习的谈论中，人们常常把机器学习能够做到的事情和他们期望它做到的事情混为一谈。实际上，机器学习是使用算法从原始数据中提取信息，并用某类模型表示信息的一门学问。我们使用模型从那些未建模的数据中推断某些信息。神经网络是机器学习的一种模型，已经存在至少50年了。

神经网络的基本单元是节点，基本上是仿照哺乳动物大脑的神经元构建的。神经元之间的连接也是仿照生物的大脑构建的，也会随着时间（通过“训练”）进化。接下来的两章会深入探讨这些模型是如何工作的。

在20世纪80年代中期和90年代早期，神经网络在架构上取得了很多重要进展。但神经网络需要大量时间和数据才能取得好的结果，这限制了它的应用场景，磨灭了人们的兴趣。21世纪初，计算机的计算能力呈指数级增长，这个行业经历了之前从未发生过的计算技术的“寒武纪大爆发”。深度学习作为这个领域中一个强有力的竞争者，在计算能力呈爆炸性增长的十年中脱颖而出，赢得了许多重要的机器学习竞赛。这股热度在2017年依然没有消退，如今在机器学习的每个角落都能看到深度学习的身影。

本章将深入探讨深度学习的定义。本书结构精心编排，实践者可以利用本书做以下事情：

* 复习线性代数和机器学习相关的基础知识；
* 复习神经网络的基础知识；
* 学习四大主流深度网络架构；
* 使用书中的示例来尝试实现实用的深度网络变体。

希望你觉得这些内容实用且易于理解。我们先快速概览机器学习的基础知识，之后的章节还会介绍一些核心概念，以便你更好地理解本书的剩余内容。

机器如何学习

在定义机器如何学习之前，首先需要定义“学习”。日常生活中，当说到“学习”时，指的是“通过学习、经验或者接受教育来获得知识”。结合我们的主题，可以把机器学习看作使用算法从数据样本中获取其结构描述的做法。计算机学到一些关于结构的信息，这代表原始数据中的信息。结构描述是所构建模型的另一种说法，包含着从原始数据中提取的信息。我们可以使用这些结构或模型来预测未知数据。结构描述（或模型）可以呈现为多种形式，其中包括：

* 决策树
* 线性回归
* 神经网络的权重

每种类型的模型使用不同的方式将规则应用于已知数据，从而预测未知数据。决策树以树结构的形式创建一组规则，而线性模型创建一组表示输入数据的参数。

神经网络有一个所谓的参数向量，用于表示网络中节点之间连接的权重。本章稍后会介绍这类模型的细节。

**机器学习与数据挖掘**

数据挖掘已经出现几十年了，它像机器学习的许多术语一样，常被曲解或错用。基于本书内容，“数据挖掘”的实践被定义为“从数据中提取信息”。机器学习的不同之处在于，它指的是数据挖掘中用于从原始数据获取结构描述的算法。下面是对数据挖掘做法的简要概括。

* 为了学习概念：
* 需要原始数据的样本。
* 从数据选取行或实例的样本：
* 这些样本代表数据中特定的模式。
* 机器从这些数据模式中学习概念：
* 机器通过算法进行学习。

在IBM和斯坦福大学工作的人工智能领域的先驱A*rthur*S*amuel*将机器学习定义为：

在不直接针对问题编程的情况下，赋予计算机学习能力的一个研究领域。

S*amuel*开发了一款可以玩西洋跳棋的软件，它能调整策略，因为它能学到输赢的概率与棋盘上某种棋面之间的关系。这种寻找通往胜利或失败的模式，然后识别并强化成功模式的基本做法，支撑着机器学习和人工智能走到今天。

机器能够通过学习达成其自身目标，对此我们已经憧憬了几十年。这也许主要是受到现代人工智能教父S*tuart* R*ussell*和P*eter* N*orvig*的著作《人工智能：一种现代的方法》的影响，书中写道：

一个缓慢、微小的大脑，无论是生物学上的还是电子学上的，是如何感知、理解、预测和操纵比自己更大、更复杂的世界的？

这句话暗示了学习的概念受到了自然界的过程和算法的启发。图1-1以图形化的方式展示了人工智能、机器学习和深度学习之间的关系。

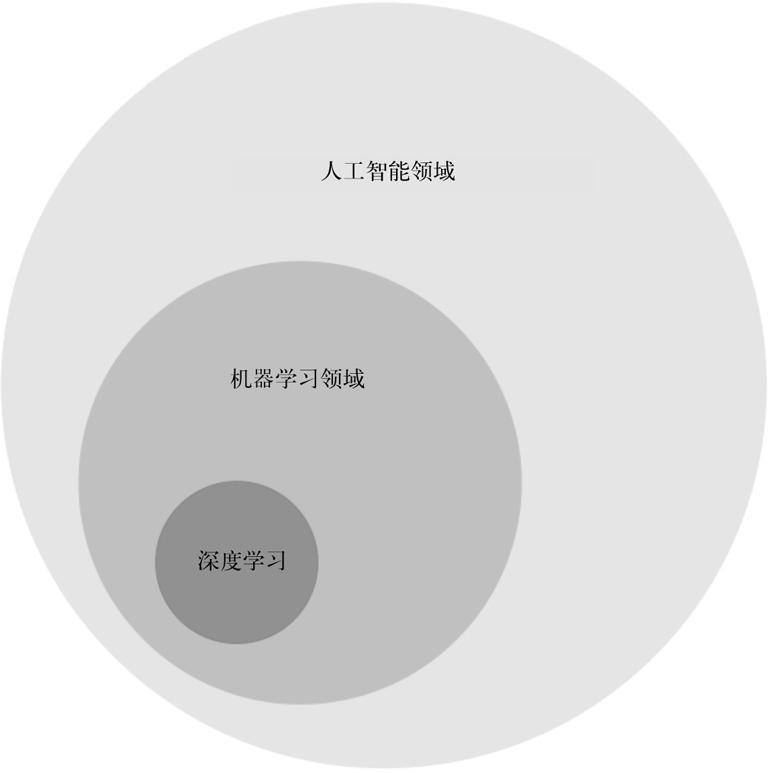


图1-1：人工智能与深度学习之间的关系

人工智能是个广阔的领域，且已经存在了很长时间。深度学习是机器学习领域的一个子集，是人工智能的子领域。下面快速了解一下深度学习的另一个根源：神经网络是如何受到生物学启发的。

生物学的启发

生物学上的神经网络（大脑）大约由860亿个神经元组成，每个神经元与许多其他神经元相连。

**人脑中全部的连接**

数据研究人员保守估计，人脑中神经元之间有超过500万亿个连接。即使今天最大的人工神经网络离这个数字也相去甚远。

从信息处理的角度看，生物学神经元是一个可兴奋的单元，可以通过电信号和化学信号处理和传输信息。生物大脑中的神经元被视作大脑、中枢神经系统的脊髓和外周神经系统的神经节的主要成分。本章稍后会介绍，人工神经网络的结构要比它简单得多。

**生物神经网络和人工神经网络的比较**

生物神经网络要比它的人工神经网络版本复杂几个数量级！

人工神经网络有两个主要特性遵循了大脑工作的一般原理。首先，神经网络最基本的单元是人工神经元（或简称为节点）。人工神经元模仿大脑的生物神经元，就像生物神经元一样，它们受到输入的刺激。这些人工神经元传递一些（但不是所有）收到的信息给其他的人工神经元，并通常伴有转换。随着本章内容的推进，将会详细讨论这些神经网络中的转换。

其次，通过训练，大脑中的神经元可以只传递那些有助于大脑达成更大目标的信号。同样，我们可以训练神经网络的神经元只传递有用的信号。本章会在这些特性的基础上，介绍人工神经网络如何通过位（*bit*）和函数来模拟生物神经网络。

**生物学对计算机科学的启发**

生物学对计算机科学的启发不限于人工神经网络。过去的50年中，学术研究还探索了自然界中带给计算机科学灵感的其他主题，例如：

* 蚂蚁
* 白蚁
* 蜜蜂
* 遗传算法

蚁群已经被研究者看作一个强大的去中心化计算机，其中没有一只蚂蚁是会导致整个系统失效的中心节点。蚂蚁不断切换任务，通过定量共识协调等元启发式算法，找到接近最优的负载均衡解决方案。蚁群能够执行清洁、防御、筑巢和觅食任务，同时根据相关需求，为每个任务分配接近最佳数量的工蚁，过程中没有个体蚂蚁直接协调工作。

什么是深度学习

对很多人来说，深度学习很难定义，因为它在过去十年中慢慢地改变了形式。一个有用的定义是，深度学习是处理“两层以上的神经网络”的技术。这个定义的存疑之处是它使深度学习听起来像是自20世纪80年代以来一直存在一样。我们认为神经网络在最近几年展现出其辉煌成果之前，就已在架构上超越了早期的网络形式（并具有更强大的处理能力）。下面是神经网络发展的一些方面：

* 比之前的网络拥有更多神经元；
* 神经网络中出现了更复杂的层/神经元之间的连接方式；
* 用于进行训练的计算能力呈现爆炸式增长；
* 自动特征提取。

基于本书的主题，深度学习被定义为具有大量参数和层的神经网络，拥有以下四种基本网络架构之一：

* 无监督预训练网络（*unsupervised* *pretrained* *network*，UPN）
* 卷积神经网络（*convolutional* *neural* *network*，CNN）
* 循环神经网络（*recurrent* *neural* *network*，RNN）
* 递归神经网络

上述架构还存在一些变体，比如混合了卷积和循环的神经网络。本书会聚焦于上面列出的四种架构的网络。

自动特征提取是深度学习相较于传统机器学习算法的另一大优点。特征提取指由网络决定数据集的哪些特征可以可靠地用于标记数据的过程。历史上，机器学习的实践者花费了他们生命中的数月、数年甚至数十年来人工创建用于数据分类的穷举特征集。在2006年深度学习大爆发开始时，最先进的机器学习算法融会了人类数十年努力积累的用于对输入分类的相关特征。对于几乎所有需要人工微调的数据类型，深度学习在精度上都超过传统算法。这些深度网络有助于数据科学团队将他们的时间、汗水和泪水节省下来，去完成更有意义的任务。

钻进奇幻的兔子洞

深度学习对计算机科学观念的渗透，超过了近代历史上的大多数技术。部分原因是这一技术不仅拥有机器学习模型中顶级的精度，而且它的创造能力甚至让非计算机科学家着迷。一个例子是艺术生成演示，即一个深层网络基于某位著名画家的作品进行训练，然后能够以这位画家的独特风格渲染其他照片，如图1-2所示。

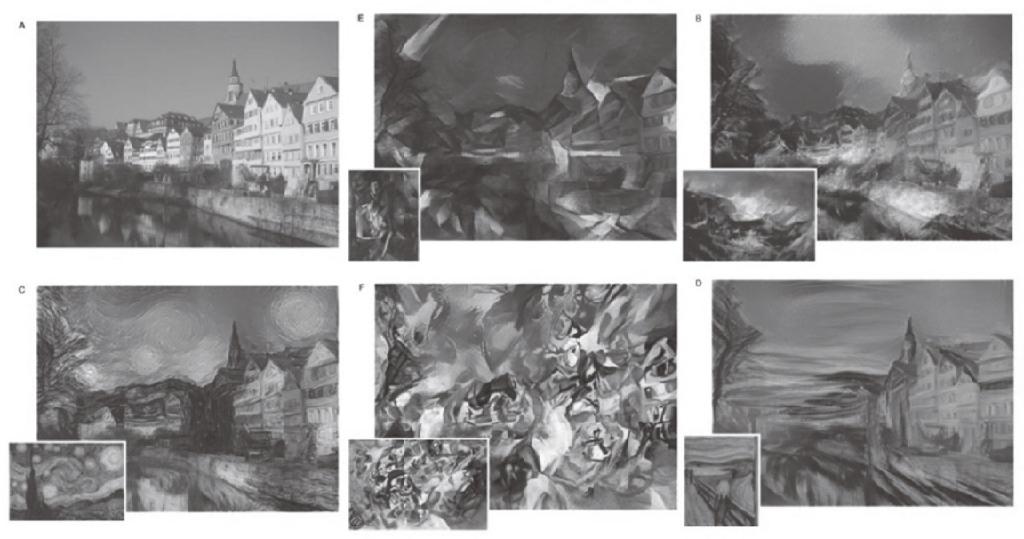


图1-2：G*atys*等人2015年的艺术风格图片

这引发了对许多哲学问题的讨论，比如“机器有创造力吗”，以及进一步的“什么是创造力”之类的问题，我们把这些问题留给你后续思考。机器学习已经发展了很多年，它就像季节变化，微妙但稳定，直到有一天你醒来时发现机器已经成为J*eopardy*节目（美国著名知识竞答节目）的冠军，或者击败了一位围棋大师。

机器能变得有智慧，并具有和人类同等的智能吗？人工智能是什么，它能变得多么强大？这些问题尚未得到解答，本书也没有所有的答案。我们只是试图展示机器智能的一些侧面。今天可以通过深度学习的实践来充实我们的生活环境。

**关于人工智能的进一步讨论**

如果想了解更多关于人工智能的信息，请阅读附录A。

# 提出问题

要想理解有关机器学习应用的基础知识，最佳方式是从提出正确的问题开始。以下是需要定义的事项。

* 我们想要从中提取信息（模型）的输入数据是什么？
* 哪种模型最适合这个数据？
* 基于这个模型，我们希望从新的数据中探索出什么样的答案？

如果能回答这三个问题，我们就可以建立一个机器学习工作流，它将建立模型并产生我们想要的答案。为了更好地完成这个工作流，首先回顾一下为了实践机器学习所需了解的一些核心概念。稍后再看看在机器学习中如何将它们结合起来，然后利用这些信息加深我们对神经网络和深度学习的理解。

# 机器学习背后的数学：线性代数

线性代数是机器学习和深度学习的基石，为求解用来建立模型的方程提供了数学基础。

线性代数的一本非常好的入门书是J*ames* E. G*entle*的M*atrix* A*lgebra*: T*heory*, C*omputations* *and* A*pplications* *in* S*tatis*-*tics*。

我们从被称为标量的基本概念开始，来了解这个领域的一些核心概念。

标量

在数学中，“标量”一词指的是向量中的元素。标量是用于定义向量空间的实数和字段元素。

在计算中，“标量”与“变量”含义相同，是与符号名配对的存储位置。这个存储位置保存着一个被称为值的未知信息量。

向量

基于使用场景，向量的定义如下：

对于正整数*n*，向量是*n*元组、有序（多）集合或者*n*个数的数组，其中的数被称为元素或标量。

详细说来，即通过一个名为向量化的过程创建一个被称为“向量”的数据结构。向量中元素的数量被称为向量的“模”（或“长度”）。向量也可以用来表示*n*维空间中的点。空间意义上，从原点到由向量表示的点的欧几里得距离就是向量的“长度”。

在数学书中，向量经常写成下面的形式：

或者

处理向量化有许多不同的方式，可以利用多个预处理步骤，得到具有不同效果的输出模型。本章稍后会详细介绍如何将原始数据转换为向量，然后第5章将更全面地讨论。

矩阵

矩阵可以看作一组向量，它们都具有相同的维度（列数）。这样矩阵就是一个二维数组，拥有行和列。

如果一个矩阵被称为*n*×*m*矩阵，那说明它有*n*行和*m*列。图1-3是一个3×3矩阵，用来展示矩阵的维度。矩阵是线性代数和机器学习的核心数据结构之一。

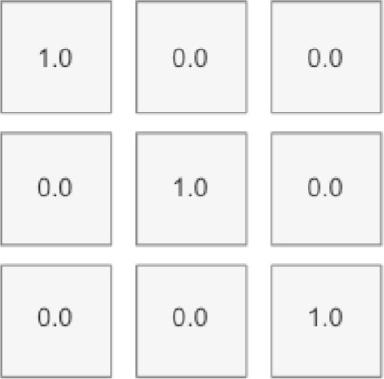


图1-3：一个3×3矩阵

张量

张量本质上是一个多维数组。向量可以看作张量的子集。

矩阵沿*y*轴的行和沿*x*轴的列延伸，每个轴是一个维度，而张量具有额外的维度。张量也有秩，相比较而言，标量的秩为0，向量的秩为1。我们也可以看出矩阵的秩为2。秩为3及以上的任何实体都被视作张量。

超平面

另一个需要了解的线性代数对象是超平面。在几何中，超平面是比其环绕空间少一维的子空间。在三维空间中，超平面有两个维度。在二维空间中，一维的线是超平面。

超平面是将一个*n*维空间分割成单独“部分”的数学构造，因此在分类等应用中会有用。优化超平面参数是线性建模的一个核心概念，在本章稍后的内容中你将体会到这一点。

相关数学运算

这一节简要回顾你需要知道的常见线性代数运算。

点积

机器学习中一个常见的核心线性代数运算是点积。点积有时被称为“标量积”或“内积”。点积计算相同长度的两个向量，并返回一个数。这是通过匹配两个向量中的元素，将它们相乘，然后将乘积相加得到的。这个计算没有直接涉及复杂的数学理论，但重要的是这个单一的数字包含了大量信息。

首先，点积是对每个向量中每个元素大小的度量。两个具有较大值的向量可以给出较大的结果，两个具有较小值的向量可以给出较小的值。当使用规范化方法从数学角度评估这些向量的相对值时，点积就是这些向量的相似度的度量。数学上把两个规范化向量的点积称为余弦相似度。

元素积

实践中经常见到的另一个线性代数运算是元素积（或哈达马积）。这个运算适用于长度相同的两个向量，它会产生一个相同长度的向量，其中每个元素的值是两个源向量相应元素的乘积。

外积

外积也被称为两个输入向量的“张量积”。取列向量的每个元素，并将其乘以行向量中的所有元素，从而在结果矩阵中创建新的一行。

将数据转换成向量

在机器学习和数据科学的工作过程中，需要分析所有类型的数据。关键的一点是能够处理各种数据类型并将其表示为向量。在机器学习中，我们使用多种类型的数据，例如文本、时间序列、音频、图像和视频。

那么为什么不能简单地把原始数据输入到学习算法中，让它处理一切呢？原因是机器学习基于线性代数，需要求解方程组。这些方程需要浮点数作为输入，所以需要一种将原始数据转换成浮点数据集的方法。稍后介绍方程组求解时会将这些概念联系起来。原始数据的一个例子是经典的鸢尾花数据集：

5.1,3.5,1.4,0.2,I*ris*-*setosa*

4.9,3.0,1.4,0.2,I*ris*-*setosa*

4.7,3.2,1.3,0.2,I*ris*-*setosa*

7.0,3.2,4.7,1.4,I*ris*-*versicolor*

6.4,3.2,4.5,1.5,I*ris*-*versicolor*

6.9,3.1,4.9,1.5,I*ris*-*versicolor*

5.5,2.3,4.0,1.3,I*ris*-*versicolor*

6.5,2.8,4.6,1.5,I*ris*-*versicolor*

6.3,3.3,6.0,2.5,I*ris*-*virginica*

5.8,2.7,5.1,1.9,I*ris*-*virginica*

7.1,3.0,5.9,2.1,I*ris*-*virginica*

另一个例子是原始文本文档：

G*o*, D*ogs*. G*o*!

G*o* *on* *skates*

*or* *go* *by* *bike*.

这两种情况都涉及不同类型的原始数据，但都需要某种程度的向量化，将数据转换为机器学习所需的形式。在某些时候，我们希望输入数据是矩阵形式，但是可以将数据转换为中间形式（如以下代码片段所示的“*svmlight*”文件格式）。我们希望机器学习算法的输入数据看起来更像序列化的稀疏向量格式*svm*-*light*，如下面的例子所示。

1.0 1:0.7500000000000001

2:0.41666666666666663 3:0.702127659574468

4:0.

5652173913043479

2.0 1:0.6666666666666666 2:0.5

3:0.9148936170212765 4:0.6956521739130436

2.0 1:0.45833333333333326

2:0.3333333333333336 3:0.8085106382978723 4:0

.7391304347826088

0.0 1:0.1666666666666665 2:1.0

3:0.021276595744680823

2.0 1:1.0 2:0.5833333333333334

3:0.9787234042553192 4:0.8260869565217392

1.0 1:0.3333333333333333 3:0.574468085106383

4:0.47826086956521746

1.0 1:0.7083333333333336

2:0.7500000000000002 3:0.6808510638297872 4:0

.5652173913043479

1.0 1:0.916666666666667 2:0.6666666666666667

3:0.7659574468085107 4:0

.5652173913043479

0.0 1:0.08333333333333343

2:0.5833333333333334 3:0.021276595744680823

2.0 1:0.6666666666666666

2:0.8333333333333333 3:1.0 4:1.0

1.0 1:0.9583333333333335

2:0.7500000000000002 3:0.723404255319149 4:0

.5217391304347826

0.0 2:0.7500000000000002

这种格式能够被快速地读入矩阵和代表标签（前例中每行的第一个数）的列向量。行内其余被索引的数字也在运行时被作为“特征”插入到矩阵中的适当位置，以便为机器学习处理过程中的各种线性代数运算做好准备。第8章将详细讨论向量化的过程。

这里有一个常见的问题：为什么机器学习算法希望数据的形式（通常）为（稀疏）矩阵？ 为了理解这一点，先快速了解一下求解方程组的基础知识。

方程组求解

在线性代数的世界中，我们对求解如下形式的线性方程组感兴趣：

其中 *A* 是由一组输入行向量构成的矩阵，*b* 是矩阵 *A* 中每个向量的标签的列向量。拿出前 一个例子中前三行序列化的稀疏输出，并将它们的值表示为如下线性代数形式：

|  |  |  |  |
| --- | --- | --- | --- |
| 列1 | 列2 | 列3 | 列4 |
| 0.7500000000000001 | 0.41666666666666663 | 0.702127659574468 | 0.5652173913043479 |
| 0.6666666666666666 | 0.5 | 0.9148936170212765 | 0.6956521739130436 |
| 0.45833333333333326 | 0.3333333333333336 | 0.8085106382978723 | 0.7391304347826088 |

这个数字矩阵是方程中的变量 *A*，每个独立的值或每行中的值被看作输入数据的一个特征。

**什么是特征**

机器学习的特征是输入矩阵 A 中任意列的值，它被用作一个独立变量。特征可以直接 从源数据中获取，但大多数情况下，要使用某些转换来将原始输入数据转换为更适合 建模的形式。 一个例子是源数据中有四个不同的文本标签的输入列。我们需要扫描所有的输入数据 并索引所使用的标签，然后根据每个标签的索引，在 0.0 和 1.0 之间规范化每行中各列 的值（0, 1, 2, 3）。这些类型的转换极大地帮助机器学习为建模问题找到更好的解决方 案。第 5 章将介绍更多向量转换技术。

我们希望找到给定行中每列的系数，用于给出输出 *b* 或每行标签的预测器函数。之前用到的序列化稀疏向量的标签如下所示：

L*abels*

1.0

2.0

2.0

前面提到的系数成为图 1-4 所示的 *x* 列向量（也称参数向量）。

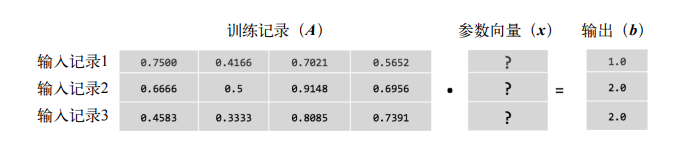


图 1-4：方程 *Ax = b* 的可视化

如果存在一个参数向量 *x*，使得该方程的解可以直接写成如下形式，则称该方程组“相容”：

从实际求解的方法的角度来解释 *x* = A-1 *b* 这个表达式很重要。该表达式仅表示解本身。变量是矩阵 *A* 的逆矩阵，它通过被称为矩阵求逆的过程来计算。考虑到不是所有矩阵都可逆，需要一种不涉及矩阵求逆的方法来解方程。其中一种方法被称为矩阵分解。一个通过矩阵分解求解线性方程组的例子是利用 LU 分解来求解矩阵 *A*。除了矩阵分解，来看一下求解线性方程组的一般方法。

**线性方程组的求解方法**

求解线性方程组有两种通用方法。第一种被称为“直接法”，它在算法上有固定的计算量。 另一种称为迭代法，通过一系列近似和一组终止条件，可以导出参数向量 *x*。当所有训练 数据（A 和 *b*）能够在一台计算机的内存中存储时，直接类的方法特别有效。使用直接法 求解线性方程组的著名例子是高斯消元法和正规方程法。

**迭代法**

当数据无法在一台计算机的主内存中存储时，迭代类的方法尤其有效，并且从磁盘中遍历各条记录使我们能够对更大的数据量建模。现在机器学习中最常见的迭代法的典型例子是随机梯度下降（*stochastic* *gradient* *descent*，SDG），本章稍后会讨论。这一领域的其他技术 还有共轭梯度和交替最小二乘法（第 3 章将详细讨论）。迭代法被证明是有效的、可扩展的方法，不仅遍历本地记录，而且将整个数据集分片到机器集群中，周期性地在所有客户端上计算参数向量的平均值，然后在每个本地建模的客户端上更新参数向量（第 9 章将详 细介绍）。

**迭代法与线性代数**

在数学层面，我们希望使用这些算法操作输入数据集。这个限制要求把原始输入数据转换成输入矩阵 *A*。这里对线性代数的快速回顾告诉我们“为什么”要不辞辛苦地将数据向量化。本书提供了将原始输入数据转换为输入矩阵 *A* 的代码示例，告诉你“如何”去做。将 数据向量化的机制也会影响学习的结果。本书稍后会介绍，在向量化之前，如何在预处理 阶段处理数据，以创建更精确的模型。

# 机器学习背后的数学：统计学

下面继续本章内容，我们回顾一下必要的统计学知识。我们需要重点掌握统计学中的一些 基本概念，例如：

* 概率
* 分布
* 似然

我们还想强调描述性统计和推断统计中一些其他的基本概念。描述性统计包括以下内容：

* 直方图
* 箱形图
* 散点图
* 平均值
* 标准差
* 相关系数

与之相反，推断统计关注如何从样本泛化到总体。下面是推断统计的一些例子：

* *p* 值
* 置信区间

概率与推断统计之间的关系：

* 概率从总体来推断样本（演绎推理）；
* 推断统计利用样本数据来推断总体特征。

在了解特定样本告诉我们的关于总体的信息之前，我们需要理解与从给定的总体中抽取样 本的不确定性。

关于一般统计学，本书不会涉及其他图书已深入介绍的广泛话题。本节目的不在于全面复 习统计学知识，而旨在带领你进入相关主题，你可以通过其他资源进行更深入的研究。免 责声明结束，下面从定义统计学中的概率开始。

概率

我们将事件 *E* 的概率定义为一个总是在 0 到 1 之间的数值。在这个背景下，值为 0 意味着 事件 *E* 不会发生，而值为 1 意味着事件 *E* 肯定会发生。很多时候，这个概率表示为浮点 数，但也可以表示为 0% 到 100% 之间的百分数，不会有低于 0% 和大于 100% 的有效概 率。例如，概率 0.35 也可以表示为 35%（0.35×100 = = 35%）。

测量概率的典型例子是抛一枚质量均匀的硬币，然后观察会出现多少次正面或反面朝 上（例如，正反面各 0.5）。样本空间的概率总为 1，因为样本空间代表给定试验的所有可 能结果。正如我们可以看到抛硬币的两个结果（“正面朝上”和它的补集“反面朝上”）， 0.5 + 0.5 = = 1，因为样本空间的总概率必须总是加起来为 1。

事件的概率表示如下：

*P(E) = 0.5*

这个表达式读作：

事件 *E* 的概率为 0.5。

**概率（*probability*）与几率（*odds*）辨析**

统计学或机器学习的初学者经常将概率和几率混淆。在继续学习之前，先搞清楚它们 的区别。

事件 *E* 发生的概率定义为：

*P(E) = (E* 发生的情况 ) / ( 全部情况 )

例如，从一副扑克牌（52 张）中抽出 A（4 张）的概率为：

4/52 = 0.077

相反，几率被定义为： (*E* 发生的情况 ) : (*E* 未发生的情况 )

现在扑克牌的例子变成了“抽出 A 的几率”：

4∶(52 – 4) = 1/12 = 0.0833333…

这里，这两个统计学概念的主要区别是分母不同（全部情况与未发生的情况）。

概率是神经网络和深度学习的中心，这归功于它在特征提取和分类这两大深度神经网络 功能中扮演的角色。如果你需要更全面地复习统计学，请参阅 B*oslaugh* 和 W*atters* 编写的 S*tatistics* *in* *a* N*utshell*: A D*esktop* Q*uick* R*eference* 一书。

进一步定义概率：贝叶斯方法与频率方法

统计学中有两个不同的流派，分别称为贝叶斯主义和频率主义。这两大流派的基本区别在于如何定义概率。

在频率方法看来，概率只在可重复测量的情况下有意义。当测量某物时，收集数据的设备的差异会导致结果有微小的变化。重复测量多次后，给定值的频率就表示测量该值的概率。

而使用贝叶斯方法时，概率的概念扩展到涵盖陈述的确定性方面。概率是我们对测量 结果的认知的陈述。对于贝叶斯方法来说，我们自己对事件的认知基本上与概率有关。

对变量估计值进行陈述之前，频率方法依赖许许多多的盲试。而贝叶斯方法处理的是 对变量的“信念”（数学术语为“分布”），并在新信息到来时更新对变量的信念。

条件概率

当我们想知道一个给定事件在另一个事件发生的前提下发生的概率时，将它表示为条件概 率，表达形式如下：

*P(E |F)*

其中 *E* 是我们对其概率感兴趣的事件。

*F* 是已发生的事件。

下面是表示一个心率正常的人在医院就诊时在 ICU 死亡概率较低的例子：

P(ICU死亡 | 非正常心率)＞P(ICU死亡 | 正常心率)

有时第二个事件 F 被叫作“条件”。在机器学习和深度学习中，条件概率很有用，因为我 们经常对何时发生多个事件以及它们如何相互影响感兴趣。使用机器学习构建分类器时， 就会用到条件概率：

*P(E |F)*

其中 *E* 是标签，*F* 是用于预测 *E* 的事实的一些属性。比如根据每位病人在 ICU 的测量数据 （这里的 *F*）来预测死亡率（这里的 *E*）。

**贝叶斯定理**

条件概率一个更常见的应用是贝叶斯定理（或贝叶斯公式）。在医学领域，它被用于计 算在一个特定疾病的检查中结果为阳性的患者实际上患有该疾病的概率。 对于任意两个事件 *A* 和 *B*，贝叶斯公式定义如下：

后验概率

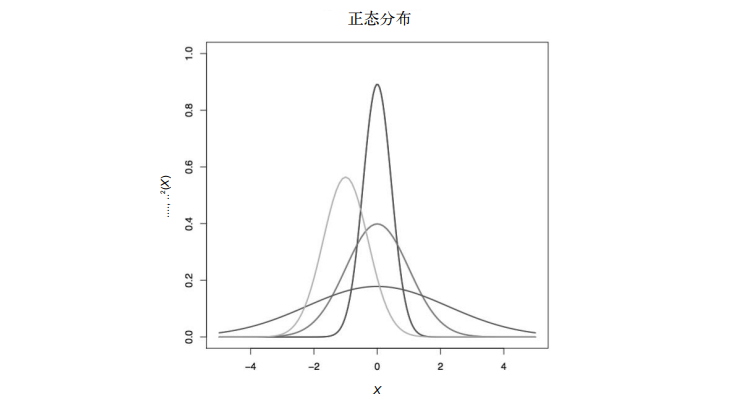
在贝叶斯统计中，考虑证据之后分配给随机事件的条件概率称为随机事件的后验概率。我们将从实验中收集的证据作为随机变量，将后验概率分布定义为依赖这个证据的未知量的概率分布。我们会在 *softmax* 激活函数（本章稍后解释）中见到这个概念的作用，其中原始输入值被转换为后验概率。

分布

概率分布是随机变量的随机结构的一个规范。在统计学中，我们依赖对数据的分布情况做出假设，来对数据进行推断。我们需要一个公式来告诉我们分布中观测值出现的频率以及分布中的点如何取值。一个广为人知的分布是正态分布（也被称为高斯分布或钟形曲线）。我们喜欢将数据集与一个分布适配，因为如果数据集合理接近分布，就可以基于这个理论上的分布来对如何操作数据做出设想。

分布分为连续型和离散型。在离散分布中，数据只能取某些值。在连续分布中，数据可以是范围内的任何值。连续分布的一个例子是正态分布，离散分布的一个例子是二项分布。

正态分布允许我们假设统计抽样分布（例如样本平均值）在指定条件下呈正态分布。正态分布（参见图 1-5），或者叫高斯分布，是以 18 世纪数学家和物理学家卡尔 • 高斯的名字命名的。正态分布由其平均值和标准差来定义，在所有变化中通常具有相同的形状。

图 1-5：正态分布的例子

机器学习中其他的相关分布包括：

* 二项分布
* 逆高斯分布
* 对数正态分布

机器学习中训练数据的分布对于理解如何向量化建模数据很重要。

**中心极限定理**

如果样本足够大，样本平均值的抽样分布近似正态分布。不管样本所属总体怎样分布，这一点都成立。

基于这一事实，可以使用基于平均值的近似正态性的检验进行统计推断。即使样本是 从非正态分布的总体中抽取的，这一点依然成立。

在计算机科学中，这个特性被应用于使用算法从非正态分布的总体中重复抽取指定大 小样本的场景。当绘制从正态分布的总体中抽取的样本的直方图时，可以看到这种特 性的作用。

长尾分布（如 Z*ipf* 分布、幂律分布以及帕累托分布）表示一种高频总体后面跟着低频总体，且整体呈逐渐减少趋势的场景。这些分布由 B*enoit* M*andelbrot* 在 20 世纪 50 年代发现，后来因作家克里斯 • 安德森的著作《长尾理论：为什么未来的商业销售更多小众商品》而广为人知。

例如，对零售商销售的商品进行排名，其中一些商品非常受欢迎，但大量商品的销量相对 较少。这种排序频度分布（主要是受欢迎程度或“销量”）经常形成幂律。从这个角度来 看，可以认为它们是长尾分布。

我们看看以下场景中的长尾分布。

* 地震损失

随着地震规模的增大，损失也越来越严重，所以损失程度不同。

* 粮食产量

有时会出现超出历史记录的产量，而模型倾向于围绕平均值调整。

* 预测从 ICU 病房出来之后死亡的概率

会有很多远远超出 ICU 病房中发生范围的、影响死亡率的事件。

这些例子在本书的分类问题背景下是相关的，因为大多数统计模型依赖于从大量数据中进 行推断。如果更有趣的事件发生在分布的尾部，并且训练样本数据中没有包含这种情况，那么模型的表现也许不可预测。这种效应会在神经网络这样的非线性模型中增强。这种情况是“样本内 / 样本外”问题的特殊情况。即使经验丰富的机器学习实践者也会惊讶于模型在不全面的训练数据样本上表现得非常好，却不能泛化到更大的数据总体上。

遵循长尾分布的事件，其实际发生的可能性是标准偏差的五倍。必须注意在训练数据中适 当选取事件，防止过拟合训练数据。稍后谈到过拟合以及第 4 章介绍调优时，将给出这种 做法的更多细节。

样本与总体

数据总体被定义为希望在实验中研究或建模的所有单元。比如将研究的总体定义为“田纳 西州所有的 J*ava* 程序员”。

数据样本是数据总体的子集，我们希望它能代表数据的精确分布，而不会引入抽样偏差 （例如对总体抽样的具体做法导致样本分布偏离）。

重采样方法

B*ootstrapping* 和交叉验证是统计学中两种常用的重采样方法，对机器学习实践者很有用。机器学习中的 B*ootstrapping* 是指从另一个样本中抽取随机样本来生成一个新样本，该样本 在每个类别的样本数之间保持平衡。在对类别高度不平衡的数据集建模时，这很有用。

交叉验证（也被称为轮换估计）是评估模型对训练数据集泛化效果的一种方法。在交叉验 证中，我们将训练数据集分成 *n* 个分片，然后将这些分片划分为训练组和测试组。使用训 练组分片进行训练，然后用测试组分片进行测试。多次轮换两组之间的分片，直到用完所 有组合。对要用到的分片数量没有强制规定，但研究人员在实践中发现 10 个分片的效果 很好。经常也会单独留存一部分数据，用作训练时的验证数据集。

选择性偏差

选择性偏差指处理的采样方法没有适当的随机化，导致样本偏斜，不能代表想要建模的总体。在对数据集重采样时，需要意识到选择性偏差的存在，这样就不会在模型中引入偏差，否则将降低模型在更大的总体上的准确度。

似然

当讨论事件发生的可能性，但没有具体提到其概率数值时，使用非正式术语似然。一般来说，使用这个术语时，谈论的是一个事件，它有一个合理的发生概率，但也可能没有。也许有一些尚未被观察到的因素也会影响事件。在非正式场合，似然也被用作“概率”的同义词。

# 机器学习如何工作

关于求解线性方程组的前一节介绍了求解 *Ax=b* 的基础知识。本质上，机器学习基于算法技术，通过优化来最小化这个方程的误差。

优化专注于改变 *x* 列向量（参数向量）中的数字，直到找到一组好的值，来得到与实际值 最接近的结果。权重矩阵中的每个权重都会在损失函数计算了由网络产生的误差（基于实 际结果，如先前所示的 *b* 列向量）之后被调整。将损失的某一部分归因于每个权重的误差 矩阵将被乘以权重本身。

本章稍后将讨论 SDG，它是实现机器学习优化的主要方法之一。随着本书内容的推进，这些 概念会与其他优化算法联结起来。也会介绍诸如正则化和学习率等关于超参数的基础知识。

回归

回归指的是试图预测实际输出值的函数，它根据自变量来估计因变量。最常见的回归类型是线性回归，它基于先前在线性方程组建模中所描述的概念。线性回归试图给出描述 *x* 和 *y* 之间关系的函数，然后对于已知的 *x* 值，准确预测 *y* 值。

**建立模型**

线性回归模型的预测是系数（来自参数向量 *x*）和输入变量（来自输入向量的特征）的线性组合。可以用下面的方程来模拟这个问题：

其中 *a* 是 *y* 轴截距，B 是输入特征，*x* 是参数向量。

这个方程可以扩展为以下形式：

线性回归求解问题的一个简单例子是根据通勤距离预测每月的汽油消耗量。在这个场景中，汽油成本是通勤距离的函数。汽油成本是因变量，而通勤距离是自变量。记录这两个量的关系，然后定义一个函数，比如：

这使得我们能够合理地基于里程预测汽油消耗量。在这个例子中，将距离作为自变量，成本是模型 *f* 中的因变量。

以下是线性回归模型的其他例子：

* 将体重作为身高的函数，以此来预测体重；
* 根据房屋的面积预测其销售价格。

**线性回归可视化**

可以将线性回归表示为寻找一条尽可能接近数据散点图中很多点的直线，如图 1-6 所示。

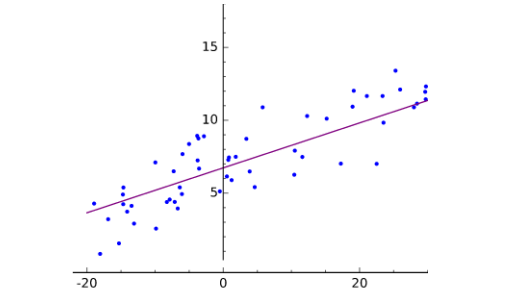


图 1-6：线性回归示意图

拟合就是定义函数 *f*(*x*)，它产生接近测量的 *y* 值或真实世界 *y* 值的 *y* 值。由 *y*=*f*(*x*) 产生的直 线接近因变量和自变量数值对的分散坐标。

**与线性回归模型联系起来**

可以将该函数与之前的方程 联系起来，其中 *A* 是要建模的所有输入示例的特征（例如，“权重”或“面积”）。每个输入记录是矩阵 *A* 中的一行。列向量 *b* 是矩阵 *A* 中所有输入记录的输出。使用误差函数和优化方法（例如 SGD），可以找到一组 *x* 参数，使得所有 预测相对于真实结果的误差最小。

如同前面所讨论的，应用 SGD 时有三个组件来求参数向量 *x*。

* 关于数据的假设

参数向量 *x* 和输入特征的内积（如上文所示）。

* 成本函数

预测的平方误差（预测 - 实际）。

* 更新函数

平方误差损失函数的导数（成本函数）。

线性回归处理直线，非线性的曲线拟合处理所有其他的曲线，尤其是那些 *x* 的指数大于 1 的曲线（这就是为什么有时机器学习被描述为“曲线拟合”）。一个绝对的拟合将穿过散点 图上的每个点。不过讽刺的是，绝对拟合通常是非常差的结果，因为这意味着模型在训练 集上训练得太完美了，如先前讨论过的，它对其未训练过的数据几乎没有预测能力（比如 不能很好地泛化）。

分类

分类模型基于某个输入特征集为输出划分类别。如果说回归给出的结果是“多少”，那么分类给出的结果是“哪一种”。因变量 *y* 是类别型而非数值型。

最基本的分类形式是二元分类，它只有一个带有两个标签的输出（两个类别，分别为 0 和 1）。输出也可以是 0.0 到 1.0 之间的浮点数，以便处理没有绝对确定性的分类。在这种情况下，需要确定一个划分两个类别的阈值（通常为 0.5）。在文献中这些分类通常被称为阳性分类（如 1.0）和阴性分类（如 0.0），1.7 节将详细讨论。

二元分类的例子包括：

* 区分某人是否患有某种疾病；
* 区分电子邮件是否为垃圾邮件；
* 区分交易是否为欺诈或虚假交易。

除了二元分类，还有具有 *n* 个类别的分类模型，我们可以对每个输出类别评分，得分最高 的类别是输出类别。当探讨具有多个输出的神经网络与具有单个输出（二元分类）的神经 网络时将进一步讨论这一点。本章稍后探讨逻辑回归以及深入探讨神经网络的完整架构 时，还会进一步讨论分类。

**推荐**

推荐是基于与用户相似的其他用户或用户以前浏览过的其他物品，向系统用 户推荐物品的过程。因 A*mazon*.*com* 而声名大噪的协同过滤就是一种著名的 推荐算法。

聚类

聚类是一种无监督学习技术，它通过计算距离，迭代地将相似的项更紧密地归类到一起。 在过程结束时，最紧密地聚集在 *n* 个中心点附近的项被认为分类到该组。K-*means* 聚类是 机器学习中一种著名的聚类算法。

欠拟合与过拟合

如同前面所提到的，优化算法首先试图解决欠拟合问题，即选取一条不能很好地逼近数据 的直线，然后使之更好地逼近数据。横切弧形散点图的直线是欠拟合一个很好的例子，如 图 1-7 所示。

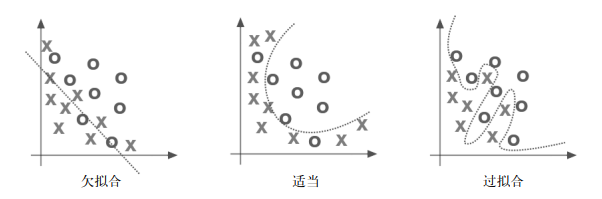


图 1-7：机器学习中的欠拟合与过拟合

如果这条线过于拟合数据，就会带来相反的烦恼，叫作“过拟合”。解决欠拟合是要优先 考虑的问题，但是在机器学习中却要花费大量的精力来避免让线过拟合数据。当说模型过 拟合数据集时，指的是它在训练数据上的误差率可能较低，但并不能很好地泛化到我们感 兴趣的数据总体。

另一种解释过拟合的方法是考虑数据的可能分布。我们试图在其中画线的训练数据集只是 更大的未知数据集的一个样本，如果需要所画的线具有预测能力，就需要它同样能较好地 拟合更大的数据集。因此，必须假设样本松散地代表更大的集合。

优化

上述调整权重以对数据做出越来越准确的猜测的过程称为参数优化。可以将这个过程看作 一种科研方法。先提出一个假设，在现实中检验它，然后不断改进或替换这个假设，以更 好地描述世界上的事件。

每一组权重都代表一个关于输入意味着什么的假设，即它们如何与一个标签的含义相关。 权重代表对网络输入和想要猜测的目标标签之间相关性的猜想。所有可能的权重和它们的 组合可以被描述为这个问题的假设空间。试图提出最好的假设就是一个在假设空间中搜 索，而我们使用误差和优化算法来实现。输入参数越多，问题的搜索空间就越大。学习的 大量时间要花在决定哪些参数要忽略，哪些要保留上。

**决策边界和超平面**

当提到“决策边界”时，讨论的是由线性模型的参数向量所生成的 *n* 维超 平面。

通过测量成本（即与真实数据点的距离）来用线拟合数据是机器学习的中心思想。这条 线应该大体上拟合数据，这通过最小化线与所有点的距离之和来实现。将线上的点 *x* 与其 对应的目标点 *y* 之间的距离之和调整到最小。在三维空间中，你可以想象山坡和山谷的差 距，并把算法想象成一位摸索着斜坡行走的盲人旅行者。优化算法，如梯度下降，会告诉 旅行者哪个方向是下坡，这样他就知道该往哪里走。

我们的目标是找到这样的权重，它能使网络的预测值（ ˆ *b* ，或者 A 和 *x* 的点积）与测试集 所知的真实值（*b*）之间的差异最小，正如之前在图 1-4 中所看到的。上面的参数向量（*x*） 就是要找的权重。网络的准确度是它的输入和参数的函数，而使它变准确的速度是它的超 参数的函数。

**超参数**

在机器学习中，既有模型参数，又有能够使网络更好、更快地训练的调优参 数。这些调优参数称为超参数，在使用学习算法进行训练时，它们负责控制 优化函数和模型的选择。

**收敛**

收敛指找到参数向量值的优化算法，它给出优化算法在所有训练样本中可能 出现的最小误差。在优化算法尝试几个不同的参数之后，就称它在该解上迭 代“收敛”。

以下是机器学习优化中的三个重要概念。

* 参数

转换输入以帮助确定网络推断的分类。

* 损失函数

在每次迭代过程中，评估分类（最小化误差）的效果。

* 优化函数

引导它走向最小误差点。 接下来仔细研究一个优化的子类——凸优化。

凸优化

在凸优化中，学习算法处理凸成本函数。如果 *x* 轴代表权重，*y* 轴表示成本，那么在 *x* 轴 上某一点处的成本将下降到 0，而当权重在两个维度上偏离其理想值时，两侧的成本呈指 数式上升。

图 1-8 显示了也可以反过来考虑成本函数

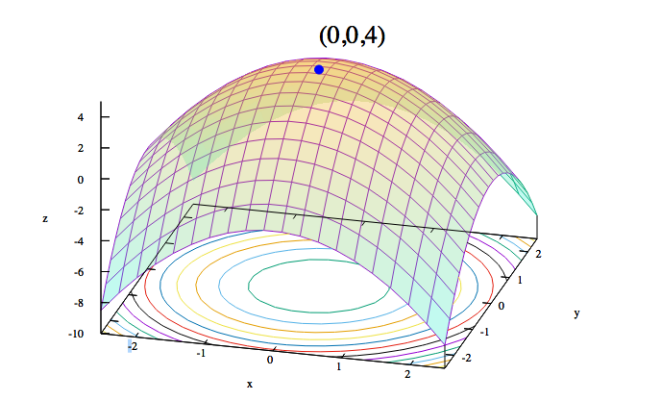


图 1-8：凸函数的可视化

另一种将参数与数据关联的方法是最大似然估计（*maximum* *likelihood* *estimation*，MLE）。 MLE 研究一条抛物线，其开口向下，纵轴为似然值，横轴为参数。抛物线上的每一点代表 给定一组参数时数据的似然值。MLE 的目标是对可能的参数进行迭代，直到找到使给定数 据最有可能的集合。

从某种意义上说，最大似然和最小成本是同一枚硬币的两面。计算两个权重相对于误差的 成本函数（这使我们处于三维空间），就像是一张固定了每个角的床单，中间下垂向外凸 出——一种碗形的函数。这些凸曲线的斜率提示算法下一步该朝哪个方向走，正如我们在 接下来探讨的梯度下降优化算法中所看到的一样。

梯度下降

在梯度下降的场景中，可以把网络预测的质量（权重 / 参数值的函数）想象为一幅风景画。 小山代表预测误差很大的位置（参数值或权重），山谷表示误差较小的位置。我们选择风 景画上的一个点作为初始权重。可以基于领域知识选择初始权重（如果正在训练一个对花 卉进行分类的网络，那么花瓣的长度重要，但颜色不重要）。假如让网络做所有工作，我 们就可以随机选择初始权重。

我们的目的是尽可能快地把权重降到误差较小的区域。像梯度下降这样的优化算法可以计 算出山坡对于每个权重的实际坡度，即它知道哪个方向向下。梯度下降法测量坡度（由权 重变化引起的误差变化），并将权重朝山谷底部移动一步。它通过求损失函数的导数来求 梯度。在优化算法中，梯度给出算法下一步移动的方向，如图 1-9 所示。

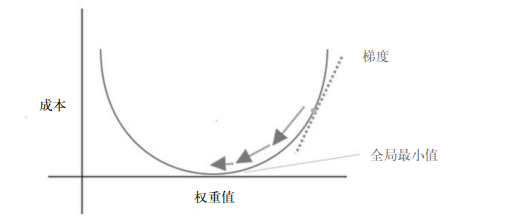


图 1-9：SGD 中朝全局最小值移动时的权重变化

导数用来衡量一个函数的“变化率”。在凸优化中，寻找函数导数等于 0 的点。这个点也 被称为函数的驻点或最小点。在优化时，考虑以最小化函数（反转成本函数的外部）的方 式优化函数。

**什么是梯度**

梯度被定义为一维函数的导数在多维函数 *f* 上的泛化，表示为函数 *f* 的 *n* 个偏导数的向 量。这对优化很有用，因为在该函数最大增长率方向上的梯度值相当于图中该方向上 的斜率。

梯度下降使用导数计算损失函数的斜率，读者应该很熟悉“导数”这个微积分概念。 在二维损失函数上，导数是抛物线上任意点的正切，也就是 *y* 的变化除以 *x* 的变化， 上升除以前进。

正如从三角学中所知，正切是一个比率，它等于直角三角形的对边（它测量垂直变化） 除以邻边（它测量水平变化）。

曲线的一个定义是一条斜率不断变化的线，线上每个点的斜率等于紧贴该点的正切。 由于斜率是由两个点求出的，那么如何精确地找到曲线上一个点的斜率呢？通过计算 曲线上距离很小的两点之间连线的斜率，然后慢慢缩小距离直到接近零来求导数。在 微积分运算中，这叫作极限。

不断重复计算误差并在低误差的方向修改权重的过程，直至权重到达误差最低的点，即准 确度最高的点。使用凸损失函数（通常在线性建模）时，损失函数只有一个全局最小值。

可以从求参数向量 *x* 的三个组件的角度来思考线性建模。

* 一个关于数据的假设，例如“在模型中用于预测的方程”。
* 成本函数，也称损失函数，例如“误差平方和”。
* 更新函数；我们取损失函数的导数。

我们的假设是学习参数 *x* 和输入值（特征）的组合会给出一个分类或实际的输出值（回 归）。成本函数告诉我们离损失函数的全局最小值有多远，我们用损失函数的导数作为更 新函数来改变参数向量 *x*。

损失函数的导数给出 *x* 中的每个参数需要调整的幅度，以便更接近损失曲线上的 0 点。本 章稍后将更详细地研究这些方程，展示它们如何用于线性回归和逻辑回归（分类）。

然而在其他非线性问题中，并不总是能够得到这样一条干净的损失曲线。这些非线性的、 假想的风景画的问题是：可能有多个山谷，但是通过梯度下降来求更低权重的机制却无法 知道它是否已经到达最低谷，或者仅仅是在一个较高的山谷的最低点。最低谷的最低点被 称为全局最小值，而其他所有山谷的最低点称为局部最小值。梯度下降法有可能陷入局部 最小值，这是它的一个缺点。第 6 章探讨超参数和学习率时将介绍解决这一问题的方法。

梯度下降面对的第二个问题是非规范化特征。本书中的“非规范特征”指需要用差别很大 的比例尺来测量的特征。如果有一个维度是百万级的，而另一个维度是小数级的，那么梯 度下降将很难找到最陡峭的斜率来使误差最小化。

**规范化处理**

第 8 章将深入研究向量化语境下的规范化方法，并介绍一些更好地解决这一 问题的方法。

SGD

在梯度下降中，在计算梯度和更新参数向量之前，计算所有训练样本的总损失。而在 SGD 中，在每次训练样本之后计算梯度和更新参数向量。这已被证明可以加快学习速度，同时 也有利于并行，本书后面详细讨论。SGD 是“全批量”梯度下降的近似值。

**小批量训练与SGD**

SGD 的另一个变体是使用多个训练样本计算梯度，但不会使用整个训练数据集。这种变体 被称为小批量 SGD 训练，它已被证明比仅使用单个训练实例的做法性能更好。应用小批 量 SGD 也会使收敛更平滑，因为每次迭代都会使用更多的训练样本来计算梯度。

随着小批量大小的增加，计算出的梯度会更接近整个训练集的“真实”梯度，这也带来了 更高的计算效率。如果设置的小批量过小（例如，一条训练记录），就无法有效地使用硬 件，尤其是在像 GPU 可用这样的情况下。相反，如果小批量过大（超过一定界限）就会 导致低效，因为可以用普通的梯度下降方法和更少的计算量（在某些情况下）求得同样的 梯度。

拟牛顿优化方法

拟牛顿优化方法是迭代算法，涉及一系列的“线性搜索”。它与其他优化方法的区别在于 搜索方向的选择方式。本书后面几章将进一步探讨这些方法。

雅可比矩阵和海森矩阵

雅可比矩阵是一个包含函数对各向量的一阶偏导数的 *m*×*n* 矩阵。

海森矩阵是一个函数的二阶偏导数的方阵，描述拥有许多变量的函数的局部曲率。海森矩阵在使用牛顿型方法解决大规模优化问题中得到了应用，因为它们是局部泰勒展开式的二次项系数。在实践中，海森矩阵很难计算，我们往往使用拟牛顿算法近似地 替代海森矩阵。这类拟牛顿优化算法的一个例子是 L-BFGS，第 2 章将详细介绍。

本书不会过多提及雅可比矩阵和海森矩阵，但是希望你能了解它们及其在更广阔的机 器学习领域中的地位。

生成模型与判别模型

使用不同类型的模型会生成不同类型的输出。两个主要的模型类型是生成模型和判别模型。生成模型根据数据的创建方式生成相应类型的响应或输出。判别模型不关心数据的创 建方式，只是简单地根据给定的输入信号给出分类或类别。判别模型侧重于对类别之间的 边界建模；与生成模型相比，它能够更细微地描述边界。判别模型通常用于机器学习中的 分类。

生成模型学习联合概率分布 *p*(*x*, *y*)，而判别模型学习条件概率分布 *p*(*y*|*x*)。分布 *p*(*y*|*x*) 是接 收输入 *x* 并产生输出（或分类）*y* 的自然分布，因此得名“判别模型”。生成模型学习分布 *p*(*x*, *y*)，被用来根据给定的输入产生可能的输出。生成模型通常被设置为捕捉数据中微妙 关系的概率图模型。

|  |
| --- |
| 指导教师意见：  指导教师签字：  年 月 日 |
| 系(教研室)意见：  主任签字：  年 月 日 |

注：此表单独作为一页。