第 3 章

深度学习与神经网络概述

深度学习越来越受欢迎的一个很好的理由是：**它具有惊人的精度**。特别是在拥有丰富的数据和足够算力的场合，深度学习是机器学习专家的首选。深度学习与传统机器学习算法的性能对比如图 3-1 所示。

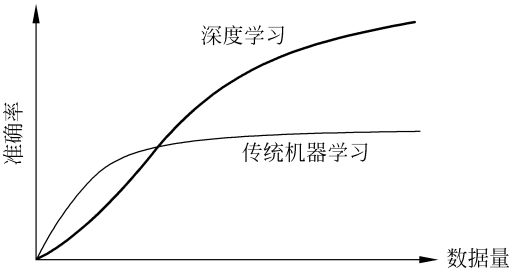


图 3-1　深度学习与传统机器学习的精度比较

深度学习是机器学习的一个分支，它通过模仿人类大脑的数据处理和模式生成能力实现自动决策。与其他机器学习算法相比，深度学习的精度明显较高，这有助于机器学习专家广泛采用深度学习算法。正是有了人工神经网络，深度学习才成为可能。人工神经网络是模拟人脑神经元的网络结构，可以使用这种网络结构进行深度学习。图 3-2 表示某个具有深度学习能力的人工神经网络。

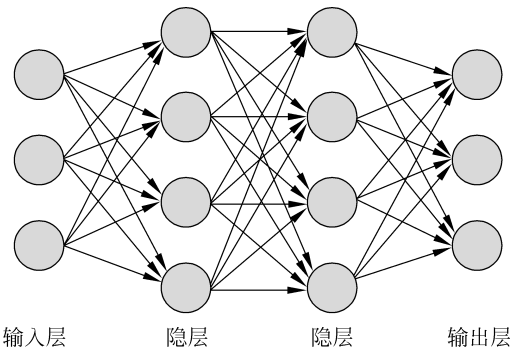


图 3-2　具有两个隐层的人工神经网络

有些人认为深度学习是一个新的发明领域，因为它推翻了其他机器学习算法。然而，关于人工神经网络和深度学习的研究可以追溯到 20 世纪 40 年代。最近深度学习的兴起主要是因为获得了大量的可用数据，更重要的是因为获得了廉价而丰富的处理能力。

本章给出深度学习的概述，介绍和讨论在深度学习中经常使用的关键概念，包括激活函数、损失函数、优化器和反向传播、正则化和特征缩放等。首先，介绍人工神经网络和深度学习的历史，以便对深度学习若干概念的根源有一定的认识，这些概念会在书中经常出现。

3.1 神经网络和深度学习研究的时间轴

神经网络和深度学习研究的时间表并不是由一系列不间断的进展构成。事实上，人工智能领域经历了几次衰退，这被称为人工智能的冬季。下面深入考察神经网络和深度学习的历史，这段历史始于 1943 年。

人工神经元的发展——1943 年，学术先驱 Walter Pitts 和 Warren McCulloch 发表了论文 *A Logical Calculus of the Ideas Immanent in Nervous Activity*，提出了一个关于生物神经元的数学模型 ——**McCulloch Pitts 神经元**。McCulloch Pitts 神经元的能力是最小的，它没有学习机制。McCulloch Pitts 神经元的重要性在于它为深度学习奠定了基础。1957 年，Frank Rosenblatt 发表了另一篇论文 *The Perceptron*：*A Perceiving and cognition Automaton*，文中介绍了具有学习和二元分类能力的**感知机**。具有革命性的感知机模型在 Mcculloch Pitts 神经元之后崛起，启发了许多人工神经网络研究人员。

反向传播——1960 年，Henry J. Kelley 发表了论文 *Gradient Theory of Optimal Flight Paths*，文中给出了一个关于连续反向传播的示例。反向传播是深度学习的一个重要概念，本章将讨论这个概念。1962 年，Stuart Dreyfus 在他的论文 *The Numerical Solution of Variational Problems* 中使用链式法则改进了反向传播方法。反向传播这个术语是由 Rumelhart、Hinton 和 Williams 在 1986 年创造的，他们使得反向传播方法在人工神经网络领域得到普遍应用。

训练和计算机化——1965 年，通常被称为“深度学习之父”的 Alexey Ivakhnenko 建立了神经网络的层次表示，并通过使用多项式激活函数成功地训练了这个模型。1970 年，Seppo Linnainmaa 提出了反向传播的自动微分方法，并编写出了第一个反向传播程序。这一发展可能标志着深度学习计算机化的开始。1971 年，Ivakhnenko 创建了一个 8 层的神经网络模型，由于其具有多层结构，因此被认为是深度学习网络。

AI 寒冬——1969 年，Marvin Minsky 和 Seymour Papert 出版 *Perceptrons*，书中猛烈抨击了 Frank Rosenblatt 的作品 *Perceptron*。这本书使得支持人工智能事业的基金极度缩水，并引发了从 1974 年到 1980 年的 **AI 寒冬**。

卷积神经网络——1980 年，Kunihiko Fukushima 介绍了 neocognitron 模型，这是第一

个 CNN，它可以识别视觉模式。1982 年，Paul Werbos 提出可以在神经网络中使用反向传播来最小化误差，人工智能社区广泛采纳了这一建议。1989 年，Yann LeCun 使用反向传播训练 CNN 识别 MNIST(Modified National Institute of Standards and Technology)数据集中的手写数字。第 7 章提供了一个类似的案例研究。

循环神经网络——1982 年，John Hopfield 引入了 Hopfield 网络，这是循环神经网络的早期实现。循环神经网络是最适用于序列数据的革命性算法。1985 年，Geoffrey Hinton、David H. Ackley 和 Terrence Sejnowski 提出了玻耳兹曼机(Boltzmann Machine，BM)，这是一种没有输出层的随机 RNN。1986 年，Paul Smolensky 提出了一种新的玻耳兹曼机器，它在输入层和隐层中没有层内连接，被称为受限玻耳兹曼机(Restricted Boltzmann Machine，RBM)。受限玻耳兹曼机在推荐系统领域得到成功的应用。1997 年，Sepp Hochreiter 和 Jürgen Schmidhuber 发表了论文，介绍了一种改进的 RNN 模型：长短期记忆(Long Short-Term Memory，LSTM)，本书将在第 8 章中讨论这个模型。2006 年，Geoffrey Hinton、Simon Osindero 和 Yee Whye The 集成了几个受限玻耳兹曼机，创建了深度置信网络(Deep Belief Network，DBN)，提高了受限玻耳兹曼机的能力。

深度学习的能力——1986 年，Terry Sejnowski 开发了 NETtalk，这是一种基于神经网络的从文本到语音的系统，可以让英语文本发音。1989 年，George Cybenko 在论文 *Approximation by Superpositions of a Sigmoidal Function* 中指出，具有单一隐层的前馈神经网络**可以逼近任意连续函数**。

梯度消失问题——1991 年，Sepp Hochreiter 发现并证明了梯度消失问题，这减缓了深度学习的发展进程，使得深度学习变得不切合实际。在 20 年后的 2011 年，Yoshua Bengio、Antoine Bordes 和 Xavier Gloot 等研究人员发现，采用线性整流单元(ReLU)作为激活函数可以防止梯度消失问题的发生。

用于深度学习的 GPU——2009 年，Andrew Ng、Rajat Raina 和 Anand Madhavan 在论文 *Large-Scale Deep Unsupervised Learning Using Graphics Processors* 中推荐使用 GPU 进行深度学习，因为 GPU 的核数比 CPU 的核数多得多。这种切换减少了神经网络的训练时间，使得深度学习的应用变得更加可行。随着 GPU 在深度学习领域的使用越来越多，Google 开发了 TPU 等专门用于深度学习的 ASIC 器件，GPU 制造商也开发出相应的并行计算平台，如 Nvidia 的 CUDA 和 AMD 的 ROCm。

ImageNet 和 AlexNet——2009 年，Fei-Fei Li 启动了一个名为 ImageNet 的数据库，其中包含 1400 万张带标签图片。ImageNet 数据库的创建为面向图像处理的神经网络发展做出了贡献，因为丰富的样本数据是深度学习的一个重要组成部分。自从 ImageNet 数据库创建以来，每年都会举办一些比赛来改进关于图像处理的研究。2012 年，Alex Krizhevsky 设计了一个使用 GPU 训练的 CNN AlexNet，与之前的模型相比，模型的预测精度提高了 75%。

生成对抗网络——2014 年，Ian Goodfellow 在当地酒吧与朋友聊天时，提出构建一种新型神经网络模型的想法。这个革命性的模型是在一夜之间设计出来的，现在被称为生成对抗网络。这种模型能够生成艺术、文本和诗歌，并且可以完成许多其他创造性的任务。本书第 12 章中给出了一个关于 GAN 模型实现的案例研究。

强化学习的力量——2016 年，DeepMind 训练了一个深度强化学习模型 AlphaGo，它可以下围棋。围棋被认为是比国际象棋复杂得多的游戏，AlphaGo 在 2017 年的围棋比赛中击败了世界冠军柯洁。

图灵奖：深度学习的先驱——2019 年，三位人工智能领域的先驱 Yann LeCun、Geoffrey Hinton 和 Joshua Bengio 分享了图灵奖。这个奖项证明了深度学习对计算机科学领域的重要性。

3.2 人工神经网络的结构

在深入考察深度学习基本概念之前，先回顾一下现代深度神经网络的发展历程。现今可以轻易地找到数百层和数千个神经元的神经网络示例，但在 20 世纪中叶之前，人工神经网络这个术语甚至根本不存在。这一切都始于 1943 年问世的一个简单人工神经元——McCulloch Pitts 神经元，它只能做简单的数学计算，没有学习能力。

3.2.1 McCulloch-Pitts 神经元

McCulloch Pitts 神经元于 1943 年问世，它只能进行基本的数学运算。每个事件都被赋予某个布尔值(0 或 1)，如果事件结果(0 或 1)的总和超过某个阈值，那么人工神经元就会触发。McCulloch Pitts 神经元的 OR 和 AND 运算可视化示例如图 3-3 所示。

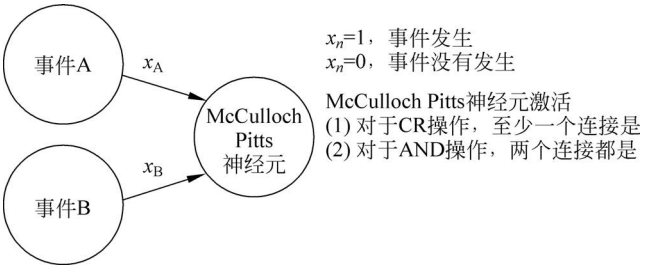


图 3-3 McCulloch Pitts 神经元的 OR 和 AND 运算

由于 McCulloch Pitts 神经元中来自事件的输入只能是布尔值(0 或 1)，所以它的计算能力是最小的。线性阈值单元(Linear Threshold Unit，LTU)的开发解决了这一限制。

1. 线性阈值单元

在 McCulloch Pitts 神经元中，每个事件的重要性都相等。因为现实世界大多数的事件不符合这种简单设置，这是有问题的。为了解决这个问题，1957 年引入了 LTU。在 LTU 中，权重被分配给每个事件，这些权重的取值可正可负。每个事件的结果仍然被赋予某个布尔值(0 或 1)，但随后与分配的权重相乘。只有当这些加权计算结果的总和为正时，LTU 才被激活。图 3-4 为 LTU 的可视化表示，LTU 是当今人工神经网络的基础。

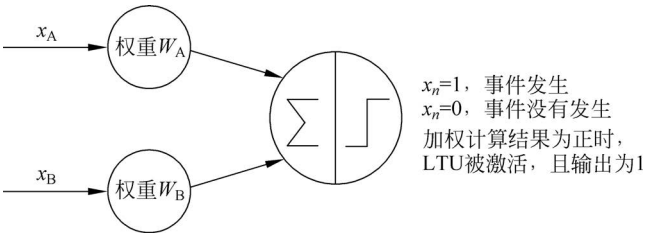


图 3-4　LTU 的可视化表示

2. 感知机

感知机是一种用于监督学习的二元分类算法，由一层 LTU 组成。在感知机中，LTU 使用与输入相同的事件输出。感知机算法可以通过调整权值来修正神经网络的行为。此外，还可以添加偏置项来提高网络模型的精度。

只有一层的感知机称为单层感知机。单层感知机有一个输出层和一个接收输入的输入层。在单层感知机中添加隐层，就可以得到多层感知机(Multi-layer Perceptron，MLP)。MLP 是一种深度神经网络，通常构建的人工神经网络就是 MLP 的具体示例。图 3-5 为单层感知机的可视化表示。

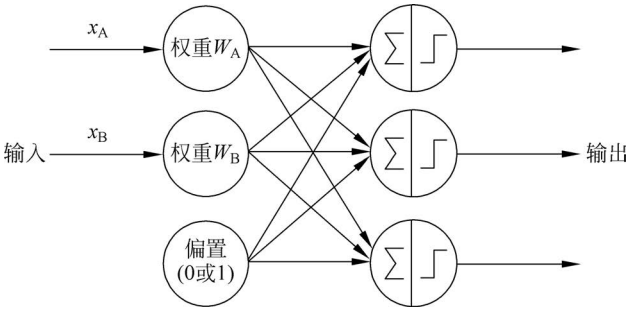


图 3-5　单层感知机的可视化表示

3.2.2　现代深度神经网络

现在使用的深度神经网络是 MLP 的改进版本。通常使用比阶跃函数(0 或 1)更加复杂的激活函数，如 ReLU、Sigmoid、Tanh 和 Softmax。现代深度神经网络通常使用梯度下

降法进行优化。图 3-6 给出了现代深度神经网络的结构。

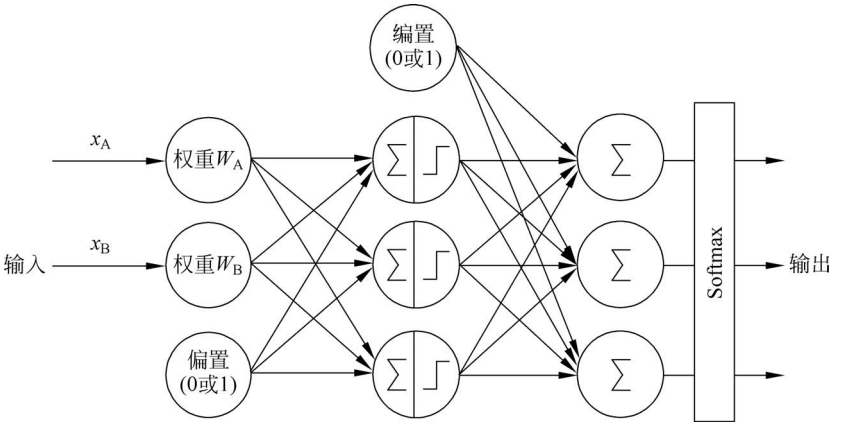


图 3-6　现代深度神经网络

已经了解了从 McCulloch Pitts 神经元到现代深度神经网络的发展历程，下面深入讨论深度学习领域中经常用到的若干基本概念。

1. 激活函数

激活函数是用来帮助人工神经网络从数据中学习复杂模式的函数。通常每个神经元的末端都有一个激活函数，它会影响神经元向下一个神经元发出的信号。换句话说，神经元的激活函数在给定一个或一组输入后确定该神经元的输出，如图 3-7 所示。

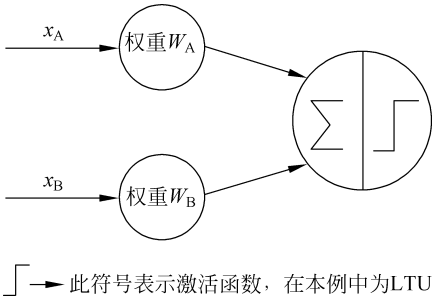
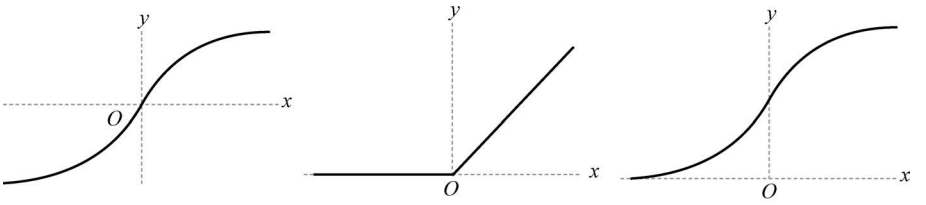


图 3-7　带有激活函数的 LTU

激活函数引入了最后一个计算步骤，为人工神经网络增加了额外的复杂性。因此，激活函数增加了所需的训练时间和处理能力。为什么要在神经网络中使用激活函数呢？答案很简单：激活函数增加了神经网络使用相关信息和抑制不相关信息的能力。如果没有激活函数，神经网络就只能进行线性变换。虽然没有激活函数可以使得神经网络模型更加简单，但会降低模型的功能，并且不能收敛于复杂的模式结构。没有激活函数的神经网络本质上只是一个线性回归模型。

可用于神经网络的激活函数有很多，常用的激活函数有 Binary Step、Linear、Sigmoid、Tanh、ReLU（Rectified Linear Unit）、Softmax、Leaky ReLU、参数化 ReLU、ELU（Exponential Linear Unit）、Swish。

其中，Tanh、ReLU 和 Sigmoid 激活函数广泛用于对单个神经元的激活。Softmax 通常用于神经网络的最后一层。图 3-8 给出了 Tanh、ReLU 和 Sigmoid 的函数图像。



(a) Tanh　(b) ReLU　(c) Sigmoid

图 3-8　Tanh、ReLU 和 Sigmoid 的函数图像

不同的激活函数通常适用于不同性质的问题。尽管 ReLU、Tanh 和 Sigmoid 函数通常在深度学习中收敛得很好，但还是应该尝试所有可能的函数来优化训练过程，以便获得尽可能高的模型精度。ReLU、Tanh 和 Sigmoid 函数之间主要区别如下。

（1）ReLU 函数是一种应用广泛的通用激活函数。ReLU 主要用于隐层。如果有死亡神经元，Leaky ReLU 可以修复潜在的问题。

（2）Sigmoid 函数比较适合分类任务。

（3）Sigmoid 函数和 Tanh 函数可能会导致梯度消失。

模型优化训练的最佳策略是从 ReLU 开始，然后尝试其他激活函数，看看性能是否有所提高。

2. 损失（代价或误差）函数

损失函数是用来衡量对于给定数据的深度学习模型性能的函数。损失函数通常基于误差，即实际（测量）值与训练模型预测值之间的差异：

$$e_i = y_i - \hat{y}_i$$

其中，e_i 为误差；y_i 为实际值；$\hat{y}_i$ 为预测值。

$$\text{误差} = \text{实际值} - \text{预测值}$$

因此，每做一次预测都会有一个误差。如果处理的是数以百万计的数据点，则需要使用一个综合函数从这些单独的误差中计算得到最终的差异，由此获得一个用于性能评估的统一值。根据以上分析，可以将这个综合函数称为**损失函数**或**误差函数**。

可以对模型的性能进行评价的损失函数有很多种，选择合适的损失函数是构建模型的一个重要步骤。这种选择必须基于问题的性质。对于需要惩罚大误差的回归问题，均方根误差是比较适合的损失函数，对于多元分类问题，则需要选择多元交叉熵。

此外，损失函数可以聚合误差项生成统一值，因此可将损失函数用作强化学习中的奖励。本书将主要使用带有误差项的损失函数，但需要注意的是，也可能将损失函数作为奖励的度量。

有多个用于深度学习的损失函数。均方根误差、均方误差、平均绝对误差（Mean Absolute Error，MAE）和平均绝对百分比误差（Mean Absolute Percentage Error，MAPE）是一些适合于回归问题的损失函数。对于二元和多元分类问题，可以使用交叉熵（对数）函数。

3. 深度学习的最优化

讨论了激活和损失函数之后，下面就可以讨论对权重和偏差的优化了。神经元和网络层中使用的激活函数对由权重和偏置项得到的线性结果进行最终的调整。可以利用这些参数（权重和偏差）进行预测。实际值和预测值之间的距离被记录为误差。使用损失函数将这些误差值聚合成单个综合值。除了这个过程之外，优化函数对权重和偏差进行小的更改，并使用损失函数衡量这些更改的效果。这个过程有助于找到最优的权重和偏差值，以最小化误差，最大限度地提高模型的准确度。这个训练周期如图 3-9 所示。

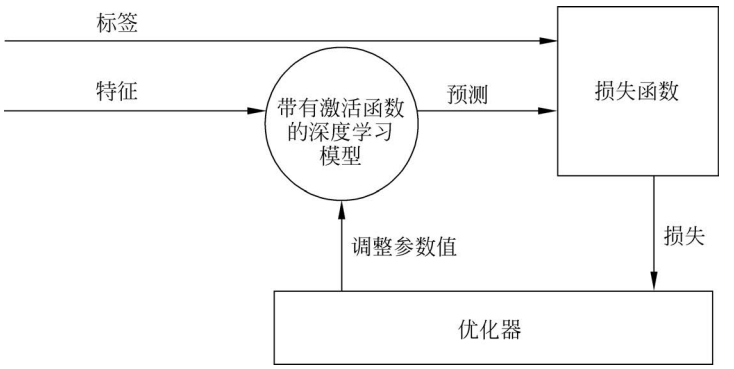


图 3-9　带有损失函数、激活函数和优化器的深度学习模型训练

4. 反向传播

优化过程会遇到多种优化算法和挑战。本节简要介绍这些优化功能和挑战。在此之前，考察一个名为反向传播的基本概念。

反向传播算法是神经网络结构中用于与优化器进行并行迭代的重要组成部分。它是神经网络学习的核心机制。propagate 的意思是传播某物。因此，backpropagation 意思是反向传输信息。这正是反向传播算法所做的工作：它将计算出的损失代回系统，优化器使用它调整权重和偏差。这个过程可以一步一步地进行，具体如下：

（1）训练后的神经网络根据当前的权重和偏差进行预测；

（2）将损失函数作为单个的误差值度量神经网络的性能；

(3) 将这个误差值反向传播到优化器,重新调整权重和偏差;

(4) 重复上述步骤,优化算法利用反向传播算法提供的信息,对神经网络中的权重和偏差进行优化。下面考察最优化算法(即优化器),它与反向传播机制并行使用。

5. 最优化算法

优化算法可以定义为一种帮助另一种算法无延迟地最大化其性能的算法。深度学习是优化算法被广泛应用的一个领域。深度学习中最常用的最优化算法包括 Adam、SGD、Adadelta、Rmsprop、Adamax、Adagrad、Nadam 等。

所有这些优化器以及前述损失和激活函数都可以在 TensorFlow 中使用。在实际应用中最常用的是 Adam 优化器和随机梯度下降(Stochastic Gradient Descent,SGD)优化器。考察所有优化器之母——梯度下降和 SGD,以便更好地理解优化算法的工作原理。

SGD 是梯度下降法的一种变体。SGD 作为一种迭代优化方法被广泛应用于深度学习。SGD 的起源可以追溯到 20 世纪 50 年代,它是最古老但比较成功的一种优化算法。

梯度下降法是一类用于最小化神经网络总损失的优化算法。它有几种实现方式。梯度下降的实现:原始梯度下降——或批梯度下降——算法在每个历元使用整个训练数据。SGD 随机选择一个观测值来衡量由于权重和偏差的变化而导致的总损失变化。最后,小批量梯度下降使用小批量数据,这样的训练仍然快速且可靠。

6. 历元

历元是超参数,表示使用训练数据集调整神经网络权重的次数。

图 3-10 表示梯度下降算法的工作原理。当机器学习专家选择更快的学习速度时,就会使用较大的步长。

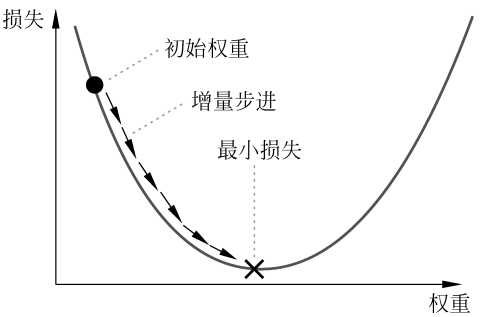


图 3-10　梯度下降的权重-损失

7. 学习率

学习率(learning rate)是优化算法中的参数,它调节每次迭代所采取的步长,使得损失函数向前移动到最小值点。学习速度较快,模型可以较快地收敛到最小值点附近,但可能会超过实际的最小值点。学习速度较慢,优化计算则可能需要较多时间。必须选择最优的学习速率,使得模型在合理的时间内找到所需的最小值点。

3.3 深度学习的优化算法

本书没有深入其他优化算法的细节，因为它们大多是梯度下降方法的某种改变或改进。因此，了解梯度下降算法就足够了。本节将考察在模型训练过程中面临对优化计算产生负面影响的挑战，以及开发了哪些优化算法来缓解这些挑战。

3.3.1 最优化面临的挑战

深度学习的最优化经常会面临局部最小值、鞍点和梯度消失三方面的挑战，下面对此进行简要讨论。

1. 局部最小值

从学习的角度看，对于神经网络的训练过程，具有单个最小值的简单权重-损失图像可能有助于形象化理解权重和损失之间的关系。然而，在实际问题中，这个图像可能包含许多局部极小值，优化算法可能收敛于某个局部极小值点而不是全局极小值点。图 3-11 表示模型如何卡在了局部最小值。

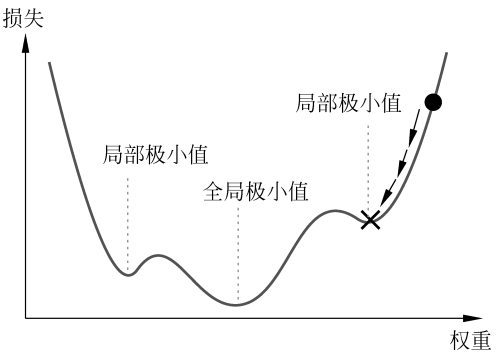


图 3-11 具有两个局部最小值和一个全局最小值的权重-损失图像

2. 鞍点

鞍点(saddle point)是图像中的稳定点，算法无法判断它是局部极小值点还是局部极大值点。鞍点两边的斜率都为零。使用多个观测值进行损失计算的优化器可能会卡在鞍点。因此，SGD 是解决鞍点问题的一种合适方法。图 3-12 表示某个带有鞍点的简化图像。

3. 梯度消失

过度使用某些激活函数(如 Sigmoid 函数)可能会对优化算法产生负面影响。因为在损失函数梯度趋近于零的时候很难降低损失函数的输出。使用 ReLU 作为隐层的激活函数是解决隐层消失梯度问题的一个有效方法。Sigmoid 函数激活函数——消失梯度问题的主要原因——及其导数如图 3-13 所示。

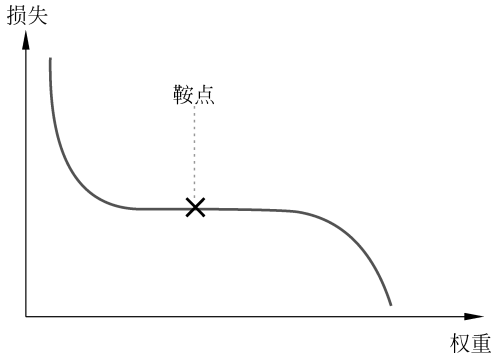


图 3-12　带有鞍点的权重-损失图像①

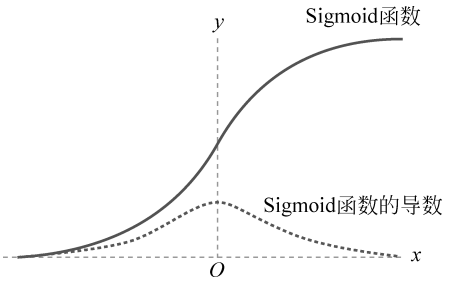


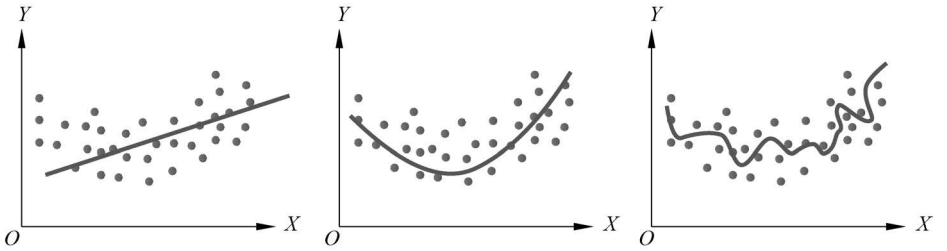
图 3-13　Sigmoid 函数及其导数

3.3.2　过度拟合与正则化

为了能够解决优化计算面临的挑战，应该尝试找到激活函数和优化函数的最佳组合，使模型能够收敛并找到理想的最小值点。

深度学习和机器学习的另一个重要概念是过度拟合。本节将讨论过度拟合问题以及如何使用正则化方法处理过度拟合问题。

第 2 章已经简要介绍了机器学习中的过度拟合。过度拟合也是深度学习面临的一个挑战。通常不希望神经网络太紧密地拟合有限的数据点集，这样会影响它在实际使用中的性能；也不希望模型不怎么适合，这样不能给出一个很好的精度。低度拟合和过度拟合问题如图 3-14 所示。



(a) 低度拟合　　(b) 特性拟合　　(c) 过度拟合

图 3-14　低度拟合和过度拟合图像

低度拟合问题的解决方案是建立一个有意义特征的良好模型，提供足够的数据，并进行足够的训练。解决过度拟合问题的正确方法是更多的数据、去除过多的特征和进行交叉验证。此外，还有一组克服过度拟合问题的复杂方法，即正则化方法。

---

① 译者注：原文为：具有两个局部最小值和一个全局最小值的权重-损失图，疑似有误。

正则化是一种对抗过度拟合的技术，一些可用的正规化方法包括提前停止、Dropout、L1 和 L2 正则化、数据扩充等。

1. 提前停止

提前停止是一种非常简单有效的防止过度拟合方法。设置足够数量的历元(训练步骤)对于达到良好精度水平是至关重要的，但可能很容易过度地训练模型，使其过于符合训练数据。如果模型在一定数量的历元后没有表现出显著的性能改善，则使用提前停止，停止学习算法。

2. Dropout

Dropout 是另一种简单有效的正则化方法。启用 Dropout 后，模型暂时从网络中移除一些神经元或网络层，这会给神经网络增加额外的噪声。这种噪声防止模型与训练数据拟合得太近，使得模型更加灵活。

3. L1 和 L2 正则化

这两种方法在损失函数中增加了额外的惩罚项，从而进一步惩罚错误。对于 L1 正则化而言，它是逻辑回归，对于 L2 正则化而言，它是脊回归。L1 和 L2 正则化在处理大量特征时特别有用。

4. 数据扩充

数据扩充是一种增加训练数据量的方法。通过对现有数据进行小的变换，可以生成更多的观察数据，并将它们添加到原始数据集。数据扩充增加了训练数据的总量，有助于防止过度拟合问题。

5. 特征缩放

深度学习的另一个关键概念是特征缩放。特征缩放是对特征范围进行归一化，使神经网络能够更准确地执行的一种方法。当特征值的范围变化很大时，一些目标函数在机器学习模型中可能无法正确工作。例如，分类器通常计算两个数据点之间的距离。当特征值的方差很大时，该特征决定了计算距离，这就意味着**该特征对结果的影响被夸大了**。缩放每个特征值的范围有助于消除这个问题。特征缩放方法有以下几种。

(1) **标准化**：调整每个特征的值，使其均值为 0 和方差为 1；

(2) **Min-Max 归一化(改变尺度)**：在[0,1]或[−1,1]之间缩放每个特征的值；

(3) **均值归一化**：从每个数据点扣除均值，并将结果除以 Max-Min。其实是将 Min-Max 做归一化处理的改进，这是一个不太流行的版本；

(4) **缩放到单位长度**：将一个特征的每个组成部分除以这个特征向量的欧氏长度。

使用特征缩放有如下两个好处。

(1) 确保每个特征对预测算法的贡献是按比例的；

(2) 加快了梯度下降算法的收敛速度，减少了模型训练的时间。

3.4　小结

本章中讨论了人工神经网络和深度学习的时间轴。这个时间轴有助于理解这些日常使用的概念如何经过多年的研究之后变成了现实。借助 TensorFlow，可以在几秒钟内将这些组件添加到神经网络模型当中。

按照深度学习的时间轴，本章详细分析了神经网络和人工神经元的结构。此外，本章还介绍了深度学习的基本概念，包括优化函数、激活函数、损失函数、过度拟合与正则化以及特征缩放等。

本章是第 2 章机器学习的基础知识的延续，在接下来的第 4 章将介绍深度学习研究中需要使用的一些流行的附加程序库，包括 NumPy、SciPy、Matplotlib、Pandas、Scikit-learn 和 Flask。