第5章

基于深度学习的分类算法

学习目标

- 了解深度学习的基本概念
- 熟悉基于深度学习的分类模型
- 掌握常用的深度学习网络

深度学习是机器学习的一个分支。相比于浅层学习方法，深层的结构使得深度学习拥有更强大的表达能力和泛化能力。近年来，随着深度学习的不断发展，基于深度学习的分类算法已成为目前主流的分类算法。本章首先深入浅出地介绍深度学习的发展历程、概念、应用以及未来发展，接着介绍 3 种常用的深度学习网络，包括卷积神经网络、循环神经网络以及长短期记忆网络，最后给出 3 个基于深度学习的案例，分别介绍如何针对图像、结构化数据和文本数据构建基于深度学习的分类模型。

5.1 深度学习概述

5.1.1 深度学习的发展历程

深度学习是机器学习的子领域，是机器学习中基于表征学习思想的技术。深度学习是人工神经网络在层数以及计算上更深层的版本，它强调从连续的层（layer）中进行学习，逐步得到原始数据的高层特征表示，并进一步用于分类等各项任务。

深度学习是从神经网络技术中逐步发展而来的，从感知机的提出到神经网络的发展，再到深度学习的兴起，深度学习的发展历史可以分为 3 个重要阶段。20 世纪 40 年代到 60 年代，感知机模型的提出带来了对人工神经网络的第一次研究热潮。当时的神经网络模型可以看作一个包含一个或多个隐含层的多层感知机，其具有更大的参数空间和更好的拟合能力。然而，这期间学术界提出的大部分神经网络模型基本上都是从神经科学角度出发的简单线性模型。同样，感知机作为一种线性分类模型，它无法学习异或函数或者解决其他非线性问题，只能在线性可分的数据上收敛。但在现实环境中，亟待处理的大部分数据并不都是线性可分的，这样的现实状况导致感知机在实际应用中存在一些局限性。

直到 20 世纪 80 年代，反向传播算法在训练神经网络中的成功应用，使得包含多个隐含层的多层神经网络的训练变得更为高效。同时，此算法的提出，也使得感知机具有解决非线性问题的能力。然而，在训练深层神经网络时，模型容易出现梯度消失和梯度爆炸等一系列问题。此外，训练深层次的神经网络需要耗费较大的算力，而当时整个研究领域的

硬件算力还略显不足，这就导致神经网络的层数一直难以增多，在许多任务中难以取得令人满意的效果。与此同时，一些新兴的机器学习算法（如支持向量机等）展现出不俗的性能表现与潜力，使得神经网络的相关研究再次陷入沉寂。

进入 21 世纪，随着各种硬件制造工艺在技术上的不断提升，硬件在算力等各方面都得到了很大的提高。同时，研究人员发现，对于多层神经网络难以训练的问题，可以采用逐层预训练的方式进行求解。算力的提升和训练策略上的突破，使得多层神经网络模型的有效训练成为可能。与此同时，多层神经网络拥有的较大参数空间和较强拟合能力也得以发挥，为机器学习和人工智能领域注入了新的活力。2012 年以后，在数据、算法和算力大力发展的基础上，深度学习时代来临了。此时，算法得以改进，大量训练样本的支持，再加上计算能力的进步，这些都使得训练深层、复杂的神经网络成为可能。同时，学术界对深度学习的研究也逐渐重视起来，新的研究成果也如雨后春笋般不断涌现。

5.1.2 深度学习的概念

深度学习中的“深度”是指从输入层到输出层经历的层的数目，即隐含层的层数。数据模型中包含的层数称为模型的深度（depth）。层数越多，表示网络训练的深度越深。深度学习的应用方式一般是端到端的形式，不需要手工设计和提取目标特征，而是通过神经网络直接处理原始数据，自动学习训练样本并输出高层特征，这让深度学习在很多特征设计较为困难的领域取得了较好的效果，并得到了广泛的应用。传统的机器学习技术，如支持向量机、逻辑回归、决策树等，它们本质上都是浅层结构算法。传统的机器学习算法对于复杂的非线性函数关系往往无能为力，如算法在样本有限的情况下表示复杂函数的能力较弱；并且针对解决复杂问题时的泛化能力也会受到制约。相较于这些浅层算法，深层神经网络具有更加强大的复杂函数的拟合能力。近年来，随着深度学习技术的快速发展，深度神经网络包含的网络结构比传统的神经网络更加多样，网络层数也更多，已经达到了数百层甚至上千层。

某种意义上，深度学习中的“学习”是相对于机器学习的“人工”而言的。在机器学习中，训练特征数据往往需要领域专家根据经验设计选取，特征设计的好坏往往直接影响机器学习的效果。在一些领域，寻找合适的特征往往是一个困难的任务。而在深度学习中，特征不再是人工设计，而是通过训练深层神经网络直接从原始数据中学习得到，即表征学习或特征学习。在训练数据足够多的情况下，深度神经网络学习到的高层特征甚至能够取得比人工设计特征更好的效果。另外，深度学习技术的另一个重要特点是分层学习。特征学习的步骤是分层执行的，即浅层的神经元负责提取简单的特征，深层的神经元将浅层特征组合起来，形成更加复杂和抽象的高层特征。这种逐层学习特征的形式也是深度学习取得良好效果的一个重要原因。

5.1.3 深度学习的应用

深度神经网络是一种统称，在实际研究过程中，随着研究任务的不同，深度神经网络模型会以不同的具体结构呈现。当前，常见的深度神经网络包括卷积神经网络（convolutional neural network，CNN）[1]、循环神经网络（recurrent neural network，RNN）[2]、长短期

记忆网络（long short-term memory，LSTM）[3] 等。其中，深度卷积神经网络具有良好的图像数据特征提取能力，在目标检测、图像分类和识别等任务上表现优异。循环神经网络以及长短期记忆网络则更适合对语音识别、文本分类等序列问题进行建模，目前在中文分词、词性标注、命名实体识别、语言翻译、文本表示等领域有着广泛的应用。

深度学习在经过不断的发展与改进之后，目前已成为人工智能领域重要的技术之一，在许多应用领域都有着出色和优异的表现，如深度学习在目标检测、图像识别、图像分类等计算机视觉领域的应用已经迅速普及；在传统机器学习领域，识别分类一个图像的标准流程是特征提取、特征筛选，最后将特征向量输入合适的分类器进行特征分类。直到 2012 年，AlexNet[4] 横空出世，它借助深度学习的算法将图像特征的提取、筛选和分类集成于一体，逐层对图像信息进行不同方向的挖掘和提取。AlexNet 在 ImageNet 图像分类大赛中以分类性能远超其他方法的成绩获得冠军，是深度学习领域中里程碑式的历史事件，一举吹响了深度学习在计算机领域爆炸发展的号角，进一步推动了基于深度学习的图像分类的研究热潮。

随着深度学习技术在计算机视觉领域获得突破性的进展，深度学习也逐渐被应用到自然语言处理领域的各项任务中。以文本为载体的信息是目前互联网时代最直接的媒体运用形式，数以亿计的用户无时无刻不在各处网络中输出大量的信息数据。文本分类技术作为信息处理的关键技术，是自然语言处理的基础任务，同时也是文本挖掘的重要基础，目前已经被广泛应用于知识挖掘和信息检索等应用领域。而将词语用向量的形式表示并输入神经网络，是深度学习应用于自然语言处理文本分类任务的基础。2013 年，谷歌提出词向量训练算法，通过大量文本数据训练得到的词向量可以应用到自然语言处理领域的各项任务中，并可以在一定程度上解决某些任务中由于缺少训练数据而导致的数据稀疏问题。随着深度学习的发展和运用，文本表示逐渐成为应用深度学习处理大规模文本分类问题的关键。由于深度学习良好的特征表达能力，越来越多的研究人员希望将该技术应用到文本表示方面，进而服务于文本处理任务，这同时成为一种新的研究趋势。

5.1.4 深度学习的未来

大数据、模型、算力是深度学习获得成功的三大支柱因素，同时也为深度学习的进一步发展和普及带来了一定的影响。一是从目前的研究现状来看，大部分领域的研究成果是以监督或者半监督的方式为基础对数据进行训练的，目前的难点就在于如何在大量无标注的数据中进行学习；二是目前的深度学习模型通常都是以大模型、大计算为支撑的，通常来说，较小规模的模型对于深度学习来说是远远不够的，无法达到深度学习中的“深”，如何得到一些比较小的模型以使深度学习技术能够在移动设备和其他各种场所中得到更广泛的应用将是一个研究问题；三是如何设计更快、更高效的深度学习算法；四是如何把数据和知识结合起来，目前，知识图谱逐渐引领了自然语言处理领域的风潮，基于深度学习的自然语言任务如何与这些知识进行融合也是一个亟待解决的问题；五是如何把深度学习的成功从一些静态任务扩展到复杂的动态决策任务中。即便深度学习还存在许多挑战，但是其正在蓬勃发展的路上，相信今后还会有更深入的突破。

5.2 卷积神经网络

本节将介绍卷积神经网络，它是计算机视觉领域普遍使用的一种深度学习模型。卷积神经网络和普通的神经网络类似，都是由可学习权重的神经元组成的。接下来会详细介绍与卷积神经网络最相关的卷积操作和池化操作。

5.2.1 卷积神经网络简介

本节以一个简单的卷积神经网络为例，介绍卷积神经网络的各个组成部分，以及它们背后的动机。

下面是一个卷积神经网络的例子。该卷积神经网络由卷积层和池化层堆叠而成，随着层次的加深，模型提取到的特征也越来越抽象、复杂。底层网络提取到的是颜色、纹理、轮廓等低层次信息，而高层网络则能够提取到高层次的图像语义信息。多层的卷积神经网络使得神经网络模型能够处理复杂的图像问题。

```
from keras import layers
from keras import models
# 构建线性堆叠的模型
model = models.Sequential()
# 为模型添加一个2D卷积层，filters表示输出空间的维度，kernel_size表示2D卷积窗口的
    # 宽度和高度，activation表示激活函数，input_shape表示输入向量大小
model.add(layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
     input_shape=(28, 28, 1)))
# 为模型添加一个空间最大池化层，pool_size表示缩小比因素
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
# 为模型添加一个2D卷积层
model.add(layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
# 为模型添加一个空间最大池化层
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
# 为模型添加一个2D卷积层
model.add(layers.Conv2D(filters=664, kernel_size=(3, 3), activation='relu'))
```

卷积神经网络相比于普通的全连接网络，能够以更小的参数量提取更多的特征。在CIFAR-10图像数据集中，一张图片是由$32\times32\times3=3072$个像素组成的。如果采用全连接层进行特征提取，每个神经元需要3072个可变参数，当输出神经元个数是1024时，需要$3072\times1024=3145728$个可变参数。随着图像规模的增大，全连接层的参数量呈爆炸式增长，这种计算无疑是对资源的浪费，过大的参数量也可能导致模型过拟合。

为了让图像能够输入全连接网络，需要将图像展平为一维向量，这使得图像丢失了原本高维空间下的分布特点。因此，需要能够处理更高维度的神经元。在卷积神经网络中，将一组提取相同区域特征的神经元称为卷积核。通过卷积核在图像不同区域的滑动提取图像的局部特征。卷积核的大小和数量将决定经过卷积运算之后输出的特征图的形状。

卷积层之后，通过ReLU非线性函数提升模型对于不同分布的泛化能力。接着通过池

化层提升模型对重要区域的捕捉能力，这里采用最大池化操作。接下来的章节会对卷积运算、非线性激活函数以及池化运算逐一进行介绍。

5.2.2 卷积运算

在计算机视觉中，图像可以看作由许多局部模型组成的一个整体。如图 5.1 所示，能将一张图像按照低层次特征分解为许多局部模式，如边缘、纹理等。这些局部模式的堆叠能够反映整体图像的特点，可以通过局部特征描述整张图像。这个重要特性使卷积神经网络具有以下两个有趣的性质。

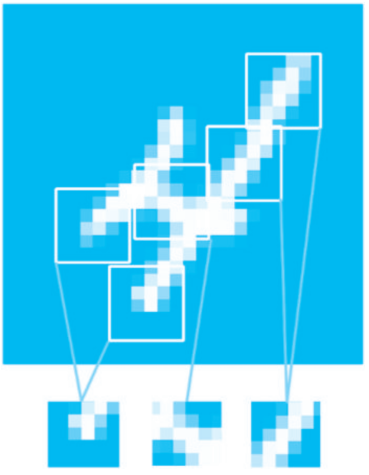

图 5.1 图像的局部模式

（1）平移不变性。卷积神经网络学习到的模式具有平移不变性。对于全连接层来说，模式具有位置敏感性，当前的局部区域移动到另一个位置时，需要重新进行训练才能够识别。而卷积神经网络则可以轻松处理局部区域平移之后发生的变化。

（2）感受野与空间层次结构。在前面的卷积神经网络示例代码中，单个卷积核的大小为 3×3，这意味着进行单次特征提取可以获得 3×3 的局部信息，将 3×3 的区域称为当前层的感受野。当进行到下一层时，同样会在特征图中 3×3 的区域进行特征提取，此时提取区域映射到原图中，就是一个 5×5 的区域。随着空间层次的加深，单个卷积核能够提取到更大模式的特征，从而使得神经网络能够学习到更加复杂、更加抽象的信息。

将输入图像经过卷积后的结果称为特征图。卷积是一个窗口滑动的过程。在特征图上固定大小的区域里，将卷积核中的对应元素与窗口区域内的元素相乘并求和，求和的值按照顺序排放。具体过程如图 5.2 所示，左边是一个 5×5 的输入矩阵，黄色部分表示卷积核作用的地方，卷积核的大小为 3×3，右侧为卷积后得到的特征图，大小为 3×3，红色区域表示黄色区域对应的卷积结果。

卷积由以下两个关键参数定义。

（1）卷积核的大小。指卷积在图中作用的图块大小，通常是 3×3 或 5×5。本例中为 3×3，这是很常见的选择。

（2）输出特征图的深度。指卷积计算的过滤器的数量。本例第一层的深度为 32，最后

一层的深度是 64。

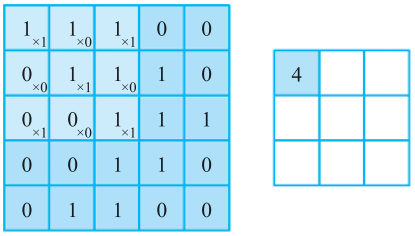

图 5.2　卷积的工作原理

在图像处理中，常用的卷积操作是二维卷积。对于图像中的每个点，二维卷积将它的值更新为周围区域的加权求和值。不同的卷积核能够起到不同的特征提取效果。在实际应用中，通常会采用多个不同大小的卷积核，旨在希望网络能够从不同方面学习到模式特征，这能极大地丰富提取模式的语义特点，从而使得模型更具有泛化性。

5.2.3　非线性激活函数

普通的卷积操作只是线性的加权求和，为了能够拟合更加复杂的场景和对象，需要在卷积层之后加上非线性的激活函数，使模型具有更强的拟合能力。在 5.2.1 节的代码示例中，采用的是 ReLU 激活函数。

图 5.3 展示了 6 种常见的激活函数，目前最常用的是 sigmoid、tanh 以及 ReLU。这些激活函数的存在使得神经网络能够应对复杂的非线性分布，同时在一定程度上起到了防止参数过多的过拟合现象。

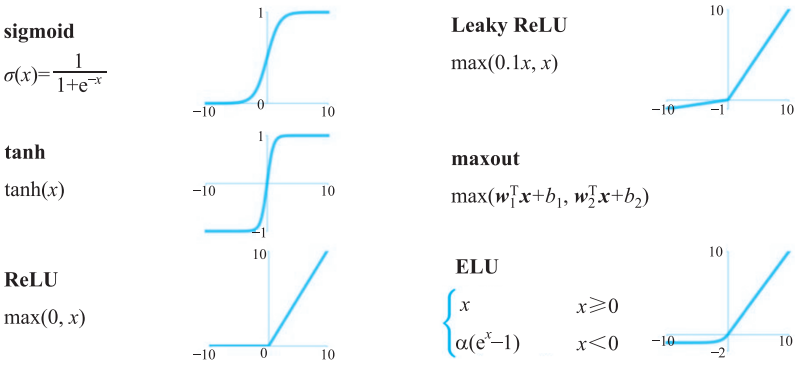

图 5.3　常用的非线性激活函数

5.2.4　最大池化运算

在许多经典卷积神经网络的示例中（包括 5.2.1 节中的示例），在每个卷积层之后都需要加上一个大小为 2×2、步长为 2 的最大池化层。如图 5.4 所示，最大池化层只会保留区域内最大的元素，通过这种操作，特征图的大小将缩减为原来的一半。

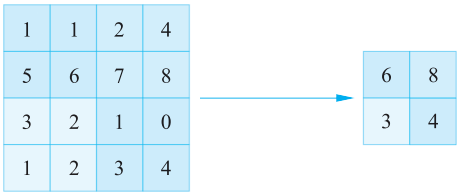

图 5.4　最大池化操作

最大池化是卷积神经网络中最常见的下采样操作。那么，为什么需要进行下采样呢？原因有以下两点。

（1）下采样操作便于模型捕捉重要信息。只保留最重要的信息，并保证最后一层特征图的每个区域能够包含整张图片的信息（或者具有全图的感受野）。

（2）防止过拟合。缺少池化层的卷积特征图过大，展平之后到全连接层的参数过多，导致严重的过拟合。

当然，除了最大池化层以外，还可以利用平均池化、步长大于 1 的卷积操作等完成下采样。在有的小型网络中，甚至可以不需要下采样操作，如文本分类任务。但是对于图像任务，需要进行大量的计算和拟合，下采样操作是不可缺少的组件。

在介绍卷积操作、非线性激活层和最大池化层之后，接下来介绍更多的深度学习模型。

5.3　循环神经网络

5.3.1　循环神经网络简介

大多数全连接神经网络仍有改进的空间：首先，在处理序列结构任务时，输出依赖于所有时刻的输入，而全连接神经网络没有考虑不同时刻的输入对输出的影响；其次，全连接神经网络要求每个样本的输入和输出的维度都是固定的，没有考虑到序列结构的数据长度是不固定的。与全连接神经网络相比，循环神经网络是一类专门用于处理时序数据的神经网络，它的每层在当前时刻会有两个输出，其中一个输出作为下一层当前时刻的输入，另一个输出作为当前层下一时刻的输入，因此循环神经网络能够考虑其他时刻对当前时刻的影响。循环神经网络是以某个时刻为单位处理数据的，它可以处理序列长度不同的数据，并且循环神经网络在每层的不同时刻的参数是共享的。循环神经网络可以看作带自循环反馈的全连接神经网络，其网络结构如图 5.5 所示，其中，x 是输入序列（长度为 T），h 是隐藏层序列，o 是输出序列，L 是总体损失，y 是目标标签序列，W_1 是输入层到隐藏层的参数矩阵，W_2 是隐藏层之间的参数矩阵，W_3 是隐藏层到输出层的参数矩阵，x_t 表示 t 时刻的输入，h_t 表示 t 时刻隐藏层状态的输出，o_t 表示 t 时刻的输出，y_t 表示 t 时刻 x_t 对应的实际标签，L_t 表示 t 时刻 x_t 对应的损失值。

图 5.5 所示的循环神经网络是一种只有一个隐藏层的 N 对 N 的神经网络。具体来说，循环神经网络输入层的输出是一个长度为 T 的序列 $x = \{x^{(1)}, x^{(2)}, \cdots, x^{(T)}\}$，这个序列对应的实际标签是一个长度为 T 的序列 $y = \{y^{(1)}, y^{(2)}, \cdots, y^{(T)}\}$，其中，$t$ 时刻输入层对应的输出和实际标签分别为 $x^{(t)}$ 和 $y^{(t)}$。循环神经网络的隐藏层在 t 时刻的输入分别是输入

层当前时刻的输出 $x^{(t)}$ 以及前一时刻 $t-1$ 的隐藏层的输出 $h^{(t-1)}$，如下所示。

$$h^{(t)} = \tanh\left(W_1 x^{(t)} + W_2 h^{(t-1)} + b_h\right) \tag{5.1}$$

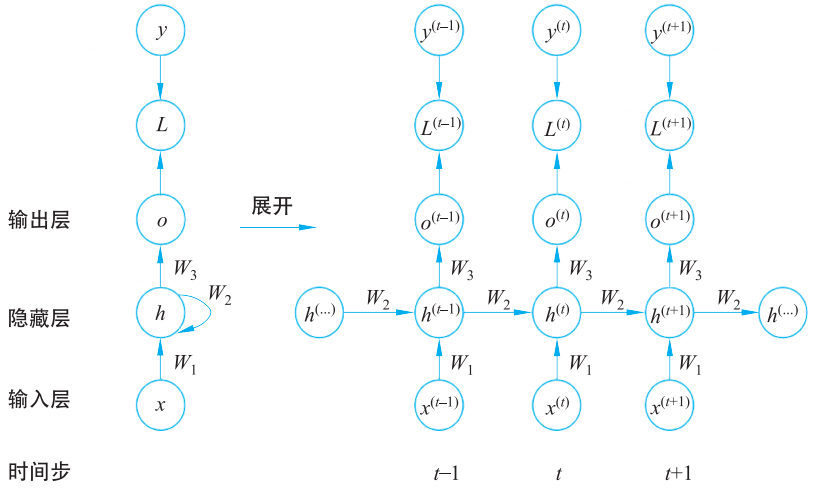

图 5.5　RNN 的网络结构

循环神经网络的隐藏层到输出层则是利用 softmax 函数将输出映射为 (0, 1) 的概率分布，具体来说：

$$o^{(t)} = \text{softmax}(W_3 h^{(t)} + b_o) \tag{5.2}$$

循环神经网络一般使用交叉熵计算某个时刻 t 在 m 个样本上的损失，整体的损失值则为所有时刻的损失之和，即

$$L^{(t)} = -\frac{1}{m}\sum_{i=1}^{m} y_i^{(t)} \log\left(o_i^{(t)}\right) \tag{5.3}$$

$$L = \sum_{t=1}^{T} L^{(t)} \tag{5.4}$$

通过上述介绍，已经了解了一个循环神经网络从输入到输出的全部过程，下面介绍如何通过代码实现一个循环神经网络。在下面这段代码中，通过堆叠一个嵌入层和多个隐藏层构建了一个循环神经网络。

```
from keras.models import Sequential
from keras.layers import Embedding, SimpleRNN
from keras.layers import Dense
# 构建线性堆叠的模型
model = Sequential()
# 添加嵌入层，100表示词典大小，64表示嵌入的维度
model.add(Embedding(100, 64))
```

```
# 添加隐藏层，64表示输出的特征大小，return_sequences表示是返回输出序列中的最后一个输
    # 出，还是返回完整序列。True表示返回完整序列
model.add(SimpleRNN(64, return_sequences=True))
model.add(SimpleRNN(64, return_sequences=True))
model.add(SimpleRNN(64, return_sequences=True))
model.add(SimpleRNN(64))
# 添加带有softmax激活函数的全连接层，46表示输出的特征大小
model.add(Dense(46, activation='softmax'))
# 编译网络，指定优化器、损失函数和评估指标
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['acc'])
model.summary()
```

用 Keras 实现的循环神经网络的输入通过 Embedding 函数设置，Embedding 函数的输入为 (T, dim_1)，T 表示输入序列的长度，其中，每个元素一般由一个长度为 dim_1 的向量表示。隐藏层 SimpleRNN 接收的输入为（dim_2, $return\_sequences$），其中，dim_2 表示每个时刻的隐藏层状态向量的维度，$return\_sequences$ 是布尔值，True 表示返回整个隐藏层对应的输出，False 表示只返回最后一个时刻对应的隐藏层对应的输出。最后，循环神经网络的输出层用函数 Dense 实现，Dense 接收的输入为（dim_3, $activation$），其中，dim_3 表示 t 时刻的输入 x_t 对应的实际标签所属的候选标签个数，$activation$ 表示激活函数，可以用 softmax、tanh 等函数实现。用 Keras 实现一个由 1 个输入层、4 个隐藏层、1 个输出层组成的循环神经网络之后，观察这个网络的参数大小。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param
=================================================================
embedding_1 (Embedding)      (None, None, 64)          6400
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, None, 64)          8256
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, None, 64)          8256
_________________________________________________________________
simple_rnn_3 (SimpleRNN)     (None, None, 64)          8256
_________________________________________________________________
simple_rnn_4 (SimpleRNN)     (None, 64)                8256
_________________________________________________________________
dense_1 (Dense)              (None, 46)                2990
=================================================================
Total param: 42,414
Trainable params: 42,414
Non-trainable params: 0
_________________________________________________________________
```

可以看到，前 3 个 SimpleRNN 层的输出都是一个 3 维张量，每个维度分别对应同时处

理的样本数量、序列的长度以及表示序列中每个元素的向量的维度，最后一个 SimpleRNN 层的输出则是一个 2 维张量，每个维度分别对应同时处理的样本数量以及表示序列中最后一个时刻元素的向量的维度。模型的参数个数为 42414 个浮点数。

5.3.2 循环神经网络的结构类型

循环神经网络可以用来处理不同的任务，处理不同任务时，模型的输入和输出有所不同，循环神经网络的结构也有所不同，主要包含以下几种情况：多对一、一对多、同步的多对多、异步的多对多。

1. 多对一

循环神经网络在处理序列分类任务，如情感分析（输出一段文字，判断这段文字表达的情感倾向）、视频分类 (输入一段视频，判断视频所属的类别）时，模型的输入是一个序列，输出是一个单独的值而不是序列。模型的输出通常可以通过两种方式得到，一种是取最后一个时刻的输出作为模型的输出，另一种是对所有时刻的隐藏层的输出取平均，如图 5.6 所示，x_t 表示 t 时刻的输入，h_t 表示 t 时刻的隐藏层状态的输出，o_t 表示 t 时刻的输出。图 5.6(a) 以最后一个时刻 t 的输出 o_t 作为模型的输出，图 5.6(b) 以所有时刻的隐藏层的输出取平均之后得到的 h 作为模型的输出。

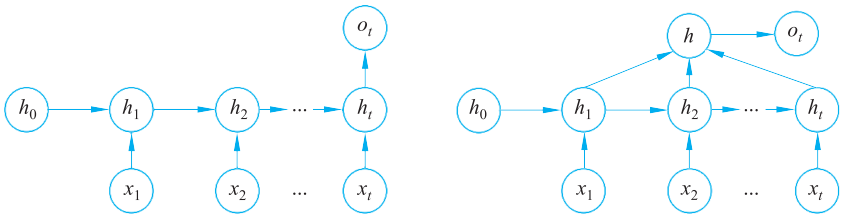

图 5.6 循环神经网络用于多对一任务

2. 一对多

循环神经网络在进行生成任务，如图例生成（输入一幅图片，生成描述这幅图片内容的文字）、特定主题下的音频或者文字生成（输入一个主题，生成属于该主题的音乐或者文字）时，模型的输入是一个值，输出是一个序列。模型的输入通常可以通过两种方式得到，一种是仅利用第一个时刻的输入作为模型的输入，另一种是所有时刻都以第一时刻的输入作为输入，如图 5.7 所示，x_t 表示 t 时刻的输入，h_t 表示 t 时刻的隐藏层状态的输出，y_t 表示 t 时刻的输出。图 5.7(a) 只以第一个时刻的输入 x_1 作为模型的输入，其他时刻的输入均为 0，而图 5.7(b) 则是所有时刻的输入都和第一个时刻的输入 x_1 相同。

3. 同步的多对多

循环神经网络在处理序列标注任务，如命名实体识别（输入一段文本，输出每个词对应的标签）时，模型的输入是序列，输出是一个等长的序列。每个时刻的输入都会对应着一个输出，如图 5.8 所示，x_t 表示 t 时刻的输入，h_t 表示 t 时刻的隐藏层状态的输出，y_t