

# 附录A

## 深度学习的基本概念

### A.1 神经网络基础

#### 1. 神经元 (Neuron)

就像形成大脑基本元素的神经元一样，这里的神经元是构成神经网络的基本结构。想象一下，当大脑得到新信息时如何处理？当大脑获得信息时，一般会处理它并生成一个输出。类似地，在神经网络中，神经元接收输入，处理它并产生输出，输出又被发送到其他神经元做进一步处理，或者作为最终结果进行输出，如图 A-1 所示。

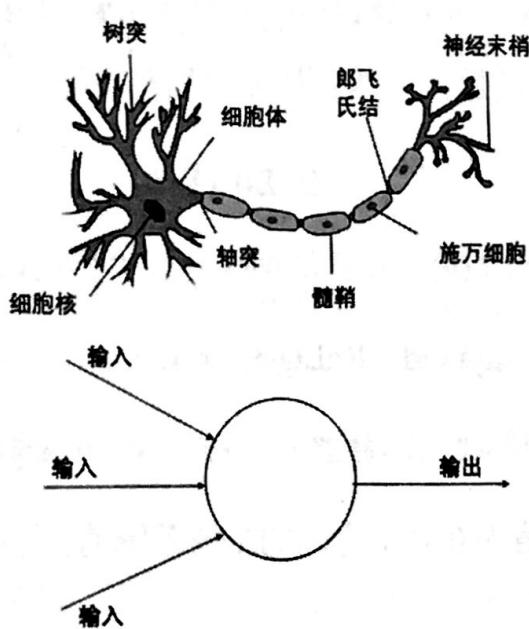


图 A-1

## 2. 权重 (Weights)

当输入进入神经元时，它会乘以一个权重。例如，一个神经元有两个输入，则每个输入将具有分配给它的一个权重。初始化时随机分配权重，并在模型训练过程中利用反向传播更新这些权重。训练后的神经网络对其输入赋予较高的权重，则认为相对而言是更为重要的输入。为零的权重则表示特定的特征是微不足道的。

假设输入为  $X_1$ ，并且与其相关联的权重为  $W_1$ ，那么在通过节点之后，输入变为  $X_1 \times W_1$ 。

## 3. 偏差 (Bias)

除权重外，另一个被应用于输入的线性分量被称为偏差。它被加到权重与输入相乘的结果中。基本上添加偏差的目的是改变权重与输入相乘所得结果的范围。添加偏差后，结果将更加接近真实值，这是输入变换的最终线性分量。

## 4. 激活函数 (Activation Function)

将线性分量应用于输入后，需要一个非线性函数，将神经元的特征保留并进行映射。激活函数将输入信号转换为输出信号。应用激活函数后的输出看起来像  $f(a \times W_1 + b)$ ，其中， $f()$  就是激活函数。

将  $n$  个输入给定  $X_1$  到  $X_n$ ，而与其对应的权重为  $W_{k_1}$  到  $W_{k_n}$ 。有一个给定值为  $b$  的偏差。权重首先乘以与其对应的输入，然后与偏差加在一起。产生的值为  $u$ 。

$$U = \sum W \times X + b$$

激活函数被应用于  $u$ ，即  $f(u)$ ，并且从神经元接收到的最终输出为  $y = f(u)$ 。

最常用的激活函数就是 sigmoid、ReLU 和 softmax。

(1) sigmoid——最常用的激活函数之一，被定义为： $\text{sigmoid}(x) = 1/(1+e^{-x})$ 。

sigmoid 变换产生一个值为 0 到 1 之间的平滑范围值。可以观察到输入值略有变化时输出值发生的变化。

(2) ReLU (整流线性单元)——最近的网络更喜欢使用 ReLU 激活函数来处理隐藏层。该函数定义为:  $f(x)=\max(x,0)$ 。当  $x>0$  时, 函数的输出值为  $x$ ; 当  $x\leq 0$  时, 输出值为 0。

使用 ReLU 函数最主要的好处是, 对于大于 0 的所有输入来说, 它都有一个不变的导数值。常数导数值有助于网络训练进行得更快。

(3) softmax——softmax 激活函数通常用于多分类问题的输出层。它与 sigmoid 函数类似, 唯一的区别就是输出被归一化为总和为 1。在一个多分类问题中, softmax 函数为每个类分配值, 这个值可以理解为概率。

## 5. 神经网络 (Neural Network)

神经网络构成了深度学习的核心, 神经网络的目标是找到一个未知函数的近似值, 它由相互连接的神经元形成。这些神经元具有权重, 并在网络训练期间根据错误来进行更新。激活函数为线性模型增加非线性因素, 并基于线性组合生成输出。

## 6. 输入/输出/隐藏层 (Input / Output / Hidden Layer)

正如它们的名字所代表的那样, 输入层是接收输入那一层, 本质上是网络的第一层。而输出层是生成输出的那一层, 也可以说是网络的最终层。处理层是网络中的隐藏层, 这些隐藏层对传入数据执行特定处理, 并将生成的输出传递到下一层。输入层和输出层是可见的, 而中间层则是隐藏的。

## 7. 多层感知器 (MLP)

单个神经元无法执行高度复杂的任务。因此, 使用堆叠的神经元生成需要的输出。在最简单的网络中, 有一个输入层、一个隐藏层和一个输出层。每个层都有多个神经元, 并且每个层中的所有神经元都连接到下一层的所有神经元。因此网络也被称为完全连接网络, 又叫多层感知器。

各层的神经元数量的选择是一个试错的过程。通常情况下, 输入层的神经元数量与输入数据的维度相同。输出层神经元的数量, 在回归问题和二元分类中通常为一个神经

元，在多分类问题中通常与类别数相同。隐藏层的神经元数量可以自由设定，通过试错找到一个最合适的价值，这通常是由通过网络的信息量决定的。在通常情况下，分类问题隐藏层的神经元数量可以设定为类别数量的 5~10 倍，回归问题隐藏层的神经元数量可以设定为输入数据维度的 2~3 倍。

## 8. 正向传播 (Forward Propagation)

正向传播是指输入通过隐藏层到输出层的运动。在正向传播中，信息沿着一个单一方向前进。输入层将输入提供给隐藏层，并生成输出，这个过程中没有反向运动。

## 9. 成本函数 (Cost Function)

当建立一个神经网络时，为了尽可能地使输出预测接近实际值，通常使用成本函数或损失函数来衡量网络的准确性，在发生错误时也利用成本函数或损失函数来惩罚网络。运行网络的目标是提高预测精度并减少误差，从而最大限度地降低成本。最优化的输出是成本函数值或损失函数值最小的输出。

## 10. 梯度下降 (Gradient Descent)

梯度下降是一种最小化成本的优化算法。就好比在爬山的时候，一步一步地走下来，而不是一下子跳下来。因此，如果从一个点  $x$  开始向下移动  $\Delta h$ ，那么位置更新为  $x - \Delta h$ ，继续保持下降，直到到达底部。

## 11. 学习率 (Learning Rate)

学习率被定义为每次迭代中成本函数中最小化的量。简单来说，下降到成本函数最小值的速率是学习率。应该非常谨慎地选择学习率，因为它不应该是非常大的，否则会错过最佳解决方案；也不应该非常小，否则网络需要花费时间进行融合。

## 12. 反向传播 (Back Propagation)

定义神经网络时，为节点随机分配权重和偏差值。一旦收到单次迭代的输出，就可以计算出网络的错误。将该错误与成本函数的梯度一起反馈给网络来更新网络的权重，以便减少后续迭代中的错误。这种使用成本函数梯度对权重的更新被称为反向传播。在

反向传播中，网络的运动是向后的，错误随着梯度从外层通过隐藏层流回，权重被更新。

### 13. 批次 (Batches)

在训练神经网络时，不是一次发送整个输入数据集，而是随机地将输入分成几个大小相等的块。与整个数据集一次性输入到网络时建立的模型相比，批量数据训练使得模型更加广义化。

### 14. 周期 (Epochs)

周期被定义为向前和向后传播中所有批次的一次训练迭代。这意味着一个周期是整个输入数据的单次向前和向后传递。更多的周期将显示出更高的网络准确性，然而，网络融合也需要更长的时间。另外，需要注意的是，如果周期数太高，网络可能会过度拟合。

### 15. 丢弃 (Dropout)

Dropout 是一种正则化技术，可防止网络过度拟合。顾名思义，Dropout 指在训练期间，网络中一定数量的神经元被随机丢弃。这意味着训练会在神经网络的不同组合的神经网络架构上进行，并将多个网络的输出用于产生最终输出，增加了网络的泛化能力。

## A.2 卷积神经网络

卷积神经网络主要应用于计算机视觉。假设有一个输入大小为  $(28, 28, 3)$  的图像，如果使用正常的神经网络，将有  $2352$  ( $28 \times 28 \times 3$ ) 个参数。并且随着图像大小的增加，参数的数量将会变得非常大，“卷积” 图像能够有效减少参数的数量。

### 1. 滤波器 (Filters)

卷积神经网络中的滤波器与加权矩阵一样，它与输入图像的一部分相乘以产生一个回旋输出。假设有一个大小为  $28 \times 28$  的图像，随机分配一个大小为  $3 \times 3$  的滤波器，并与图像不同的  $3 \times 3$  部分相乘，形成所谓的卷积输出。滤波器尺寸通常小于原始图像尺寸，一般设定为  $3 \times 3$  或  $5 \times 5$  的大小。

如图 A-2 所示，这个滤波器是一个  $3 \times 3$  的矩阵。

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

图 A-2

与图像中每个  $3 \times 3$  的部分相乘，可以形成卷积特征图，如图 A-3 所示。

输入	滤波器	特征图																																											
<table border="1"> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> </table>	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1	1	0	0	1	1	0	0	<table border="1"> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> </table>	1	0	1	0	1	0	1	0	1	<table border="1"> <tr><td>4</td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> </table>	4								
1	1	1	0	0																																									
0	1	1	1	0																																									
0	0	1	1	1																																									
0	0	1	1	0																																									
0	1	1	0	0																																									
1	0	1																																											
0	1	0																																											
1	0	1																																											
4																																													
<table border="1"> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> </table>	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1	1	0	0	1	1	0	0	<table border="1"> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> </table>	1	0	1	0	1	0	1	0	1	<table border="1"> <tr><td>4</td><td></td><td></td></tr> <tr><td>2</td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> </table>	4			2					
1	1	1	0	0																																									
0	1	1	1	0																																									
0	0	1	1	1																																									
0	0	1	1	0																																									
0	1	1	0	0																																									
1	0	1																																											
0	1	0																																											
1	0	1																																											
4																																													
2																																													
<table border="1"> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> </table>	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1	1	0	0	1	1	0	0	<table border="1"> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> </table>	1	0	1	0	1	0	1	0	1	<table border="1"> <tr><td>4</td><td>3</td><td></td></tr> <tr><td>2</td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> </table>	4	3		2					
1	1	1	0	0																																									
0	1	1	1	0																																									
0	0	1	1	1																																									
0	0	1	1	0																																									
0	1	1	0	0																																									
1	0	1																																											
0	1	0																																											
1	0	1																																											
4	3																																												
2																																													
<table border="1"> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> </table>	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1	1	0	0	1	1	0	0	<table border="1"> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> </table>	1	0	1	0	1	0	1	0	1	<table border="1"> <tr><td>4</td><td>3</td><td>4</td></tr> <tr><td>2</td><td>4</td><td>3</td></tr> <tr><td>2</td><td>3</td><td>4</td></tr> </table>	4	3	4	2	4	3	2	3	4
1	1	1	0	0																																									
0	1	1	1	0																																									
0	0	1	1	1																																									
0	0	1	1	0																																									
0	1	1	0	0																																									
1	0	1																																											
0	1	0																																											
1	0	1																																											
4	3	4																																											
2	4	3																																											
2	3	4																																											

图 A-3

## 2. 池化层 (Pooling)

在卷积层之间定期引入池化层，是为了减少一些参数，并防止过拟合。最常见的方法是使用 MAX 操作的池化层，如尺度为  $2 \times 2$  的池化层，它使用原始图像的每个  $4 \times 4$  矩阵中的最大值作为输出，有效地减少了输出参数，如图 A-4 所示。

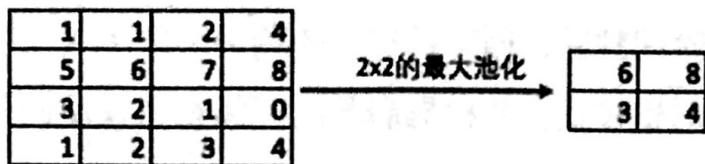


图 A-4

此外，也可以使用其他操作（如平均池化）进行池化，但是最大池化在实践中表现更好。

### 3. 填充 (Padding)

填充是指在图像之间添加额外的零层，以使输出图像的大小与输入相同。应用滤波器之后，在使用相同填充的情况下，卷积层具有与实际图像相同的大小。有效填充是指将图像保持与实际图像相同的像素数量。在这种情况下，输出的长度和宽度的大小，虽然在每个卷积层处不断减小，但是输出与原始图像相同的向量。

### 4. 数据增强 (Data Augmentation)

数据增强是指从给定数据生产新数据，这被证明对预测有益。例如，如果光线变亮，可能更容易在较暗的图像中看到猫。或者，数字识别中的 9 可能会稍微倾斜或旋转，在这种情况下，旋转将解决问题并提高模型的准确性。通过增亮或旋转，可以提高数据的质量，这被称为数据增强。

## A.3 循环神经网络

循环神经网络特别适用于序列数据，先前的输出用于预测下一个输出。隐藏神经元内的循环使它们能够存储有关前一个单元的信息一段时间，以便能够预测输出，并且隐藏层的输出在  $t$  时间戳内再次反馈到隐藏层，如图 A-5 所示。

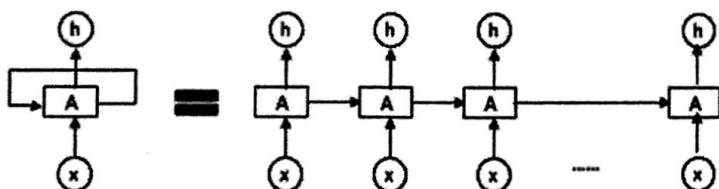


图 A-5

只有在完成所有的时间戳后，循环神经元的输出才能进入下一层。发送的输出更广泛，状态信息保留的时间也较长，并根据展开的网络将错误反向传播且更新权重，这被称为通过时间的反向传播（BPTT）。

### 1. 循环神经元（Recurrent Neuron）

循环神经元是在  $t$  时间内，将神经元的输出反馈给自身（见图 A-5），输出将返回输入  $t$  次。展开的神经元看起来像连接在一起的  $t$  个不同的神经元，这个神经元的基本优点是给出了更广义的输出。

### 2. 梯度消失问题（Vanishing Gradient Problem）

在激活函数的梯度非常小的情况下，会出现梯度消失问题。在权重乘以这些小梯度值的反向传播过程中，它们往往变得非常小，并且随着网络进一步深入而“消失”，这使得神经网络忘记了长距离依赖。但长期依赖对循环神经网络来说是非常重要的，可以通过使用不具有小梯度的激活函数 ReLU 来解决这个问题。

### 3. 梯度爆炸问题（Exploding Gradient Problem）

这与梯度消失问题完全相反，激活函数的梯度过大，在反向传播期间，它使特定节点的权重相对于其他节点的权重非常高，这使得它们不重要。这个问题可以通过剪切梯度轻松解决，使其不超过一定值。