**第3章**

# 深度学习简介

|  |  |
| --- | --- |
|  | 本 章 内 容 |
|  | ◎ 深度学习历史简介  ◎ 深度学习原理 |

## 3.1 深度学习历史简介

神经网络诞生于人类对大脑的探索，现如今的神经网络已经是一个多学科交叉的科研领域。深度学习则是多层神经网络的组合形式，其内部结构灵活，可利用繁多的数学方法及不同的数学方法组合。

神经网络的发展历程可谓崎岖，1943年两位芝加哥大学的研究人员McCullough 和 Walter Pitts借鉴了神经细胞生物原理首次提出神经元的M-P模型如图3.1，该模型是历史上的第一个神经元模型即对人类大脑工作原理的一个建模，可以看到其原理和神经网络很类似，均是对输入信号的加权求和并与阈值进行比较。

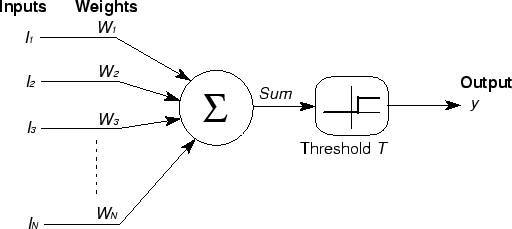


图3.1 M-P模型

1958年，Frank Rosenblatt提出了一种名为感知机的神经网络如图3.2，类似于目前神经网络中常用的全连接层，其主要作用是做二元分类，并且能够采用梯度下降法从训练样本中学习更新权重有效收敛，该模型的提出也标志着神经网络正式走进了实际应用场景之中，理论与实践的结合加快了神经网络的发展速度。

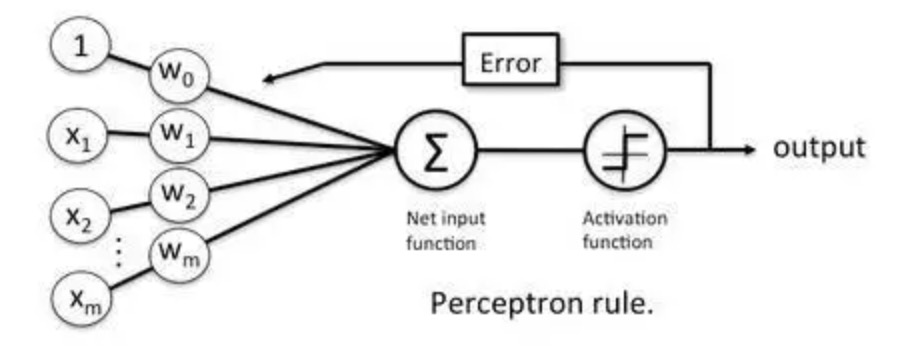


图3.2 感知机

然而神经网络的发展并不是一帆风顺。1969年之前神经网络都一直都是计算机领域热门的研究内容，Marvin Minsky 和 Seymour Papert从数学的角度证明了神经网络的局限性，只能处理线性分类问题，由于其简单的模型结构，使其连异或逻辑也无法表述，神经网络的发展也因此止步不前。

直到1983年，Hinton, G. E. 和 Sejnowski, T. J.提出了隐单元的概念，即现如今的隐藏层，隐藏层进一步提高了神经网络的灵活性，在1986年Hinton提出了适合于多层感知机的反向传播算法，并采用了sigmoid函数作为非线性映射，从而解决了非线性问题并且能有效的进行训练与参数更新，神经网络也迎来了第二次的热潮。1989年Lecun提出了目前称霸图像领域的卷积神经网络，并在手写数字的识别上取得了不错的效果。

但其后的10多年的时间内，统计学模型的理论快速发展，同期提出了支持向量机这样的通用性较好的模型，其可解释性较强，而神经网络却是黑盒模型不可解释，没有严格的数学理论支持。在1991年反向传播算法也被指出存在梯度消失的问题，导致模型没法有效的进行收敛，该发现可谓是釜底抽薪，神经网络再一次淡出了人们的视野。

直到2011年神经网络正式进入快速发展阶段。relu激活函数被提出，该激活函数能有效的解决梯度消失的问题，2012年Hinton采用了卷积神经网络在ImageNet图像分类的竞赛中打破了支持向量机的记录，可谓是完全碾压，也正是从这个时候起卷积神经网络才真正被大家所认可。

随着硬件设备的快速发展，GPU的强大处理能力，为神经网络铺平了一条宽大的道路。神经网络在ImageNet图像分类的竞赛中，准确率逐年提升。现今，深度学习已经不仅在图像方面独具一头，在很多其它领域也都取得了突破性进展，包括语音识别、自然语言处理、人脸识别、机器翻译等等。在未来，基于统计的模型还会有什么样的突飞猛进我们不得而知，但是深度学习一定会是这些模型中不可小觑的中坚力量。

## 3.2 深度学习原理

**1.什么是深度学习**

深度学习是机器学习领域内的一个分支，其主要目的在于给定一组数据，学习出数据的有用表述，即有意义的数据映射关系。在上一节我们提到过神经网络最大的缺点即是其结构过于简单，没法表述复杂的模型，从理论上来说，模型的参数如果越多，那么其表述能力就会越强，这就意味着模型能学习更复杂的任务。深度学习就是在神经网络的基础上进行了多层的叠加，强调从连续的层中进行学习，每一层的任务皆是在上一层的基础上进行进一步的特征提取。那么在每一层的神经网络中又到底是这么运作的呢，我们一起来看下这个工程。

通常深度学习由三部分构成，输入层，隐藏层，输出层。其中输入层即待输入的数据，输出层即我们需要的结果，隐藏层是对数据到结果的一个关系映射，其可能只有一两层也可能有上百层。每一层的隐藏层所做的操作其实就是学习到一个权重也被称为参数，该权重能让输入数据正确的计算出输出数据，学习的过程就是找到一个合适的权重的过程，如图3.2-1所示。但是如何才能找到合适的权重呢，特别是当隐藏层数量很多的时候，此时可能有成百上千甚至更多的参数需要计算，不同的参数其对结果的影响也可能大不相同，因此寻找合适的权重是一项很重要的任务。

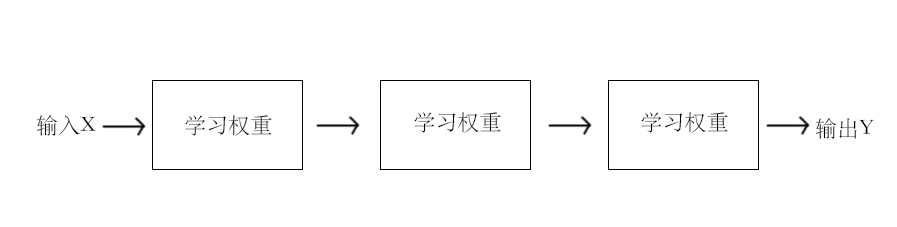


图 3.2-1 权重学习

**2.损失函数**

如果想要得到正确的输出结果，首要方法肯定是把我们的计算结果与真实的结果进行比较，并不断缩小两者的差距，那么该怎么衡量预测值与真实值得差距呢，这就是接下来要讲解的损失函数loss function。损失函数的输入就是预测值与真实值，并采用一种能计算出两者距离值的方法来进行评估，如下图所示。

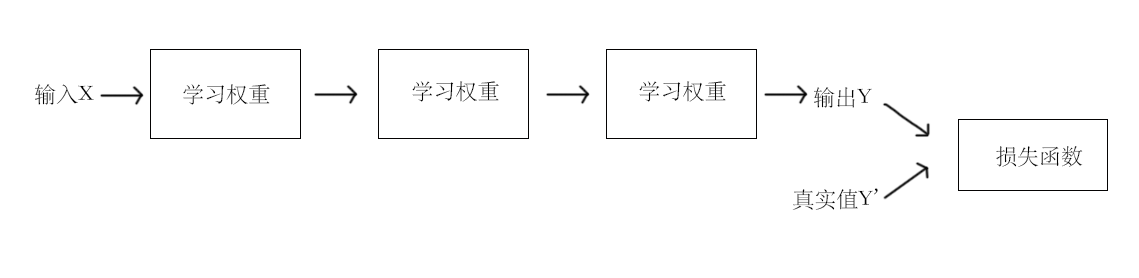


图 3.2-2 损失函数

损失函数种类很多，不同的任务通常采用的是不同的损失函数，例如回归问题通常采用的就是平方损失函数。

对于分类问题来说通常会采用对数损失函数或对数似然损失函数。

当然损失函数也可以采用自定义的方法，只要能合理计算出真实值与预测值的距离，并且可微。有了损失函数，只需要最小化损失函数的值，并采用反向传播的方法即可更新权重，从而寻找到最适合当前模型的参数，反向传播的原理我们将在后续章节进行讲解。

**3.激活函数**

对于一些非线性的任务或者说较为复杂的任务，例如图像识别，这些任务如果只是采用简单的线性变化是无法完成的，对于一个深度学习的模型来说，不管其多深，在没有激活函数的前提下永远都是一个线性函数，这和一层网络是没有区别的，而正是激活函数的出现，给神经网络带来了新的生命。激活函数很多，下文为读者介绍常用的三种激活函数。

**3.1 sigmoid函数**

sigmoid函数是最早出现的激活函数，通常会采用来表示，其公式如下

sigmoid函数的定义域是整个实数集,它可以把结果映射到一个(0,1)的区间，如图3.2-3所示。由于其结构简单，单调连续并且容易求导，所以在机器学习领域应用很广泛，逻辑回归其实就是一层神经网络加上sigmoid激活函数。但是sigmoid有一个致命的缺点，当输入值过大或者过小时，其导数的值接近于0，当网络层数较多时就会出现多个接近于0的值相乘，这就导致了梯度消失的问题模型的收敛速度就会下降，性能不够理想。

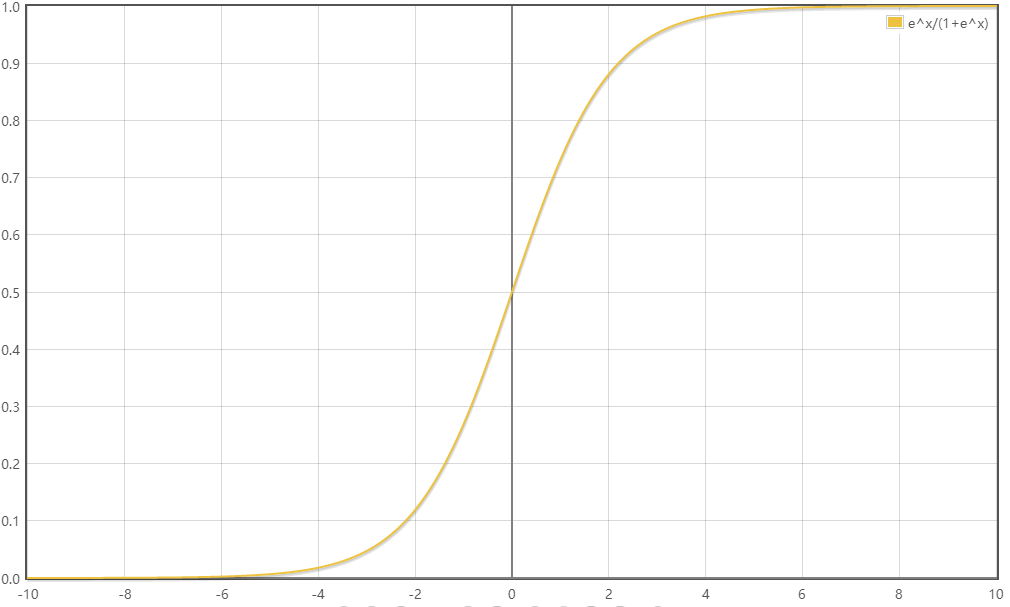


图3.2-3 sigmoid激活函数

**3.2 tanh函数**

tanh函数是对sigmoid函数的拉伸平移得到的，它能把结果映射到一个

(-1,1)的区间，tanh的效果比sigmoid的效果略好一些，但缺点和sigmoid一样，其函数图像如3.2-4所示，公式如下

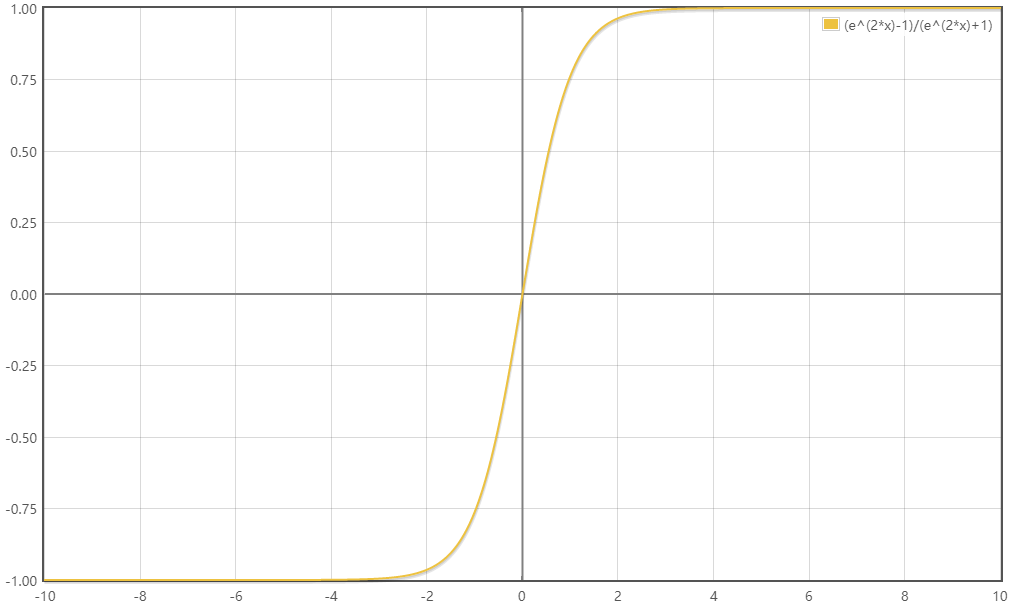


图3.2-4 tanh激活函数

**3.3 relu函数**

rule即修正线性单元，也被称为斜坡函数，relu是分段线性的，当输入为负数时，输出均为0，当输入为非负数时，输出与输入相同，对应的函数图像如3.2-5所示，公式如下

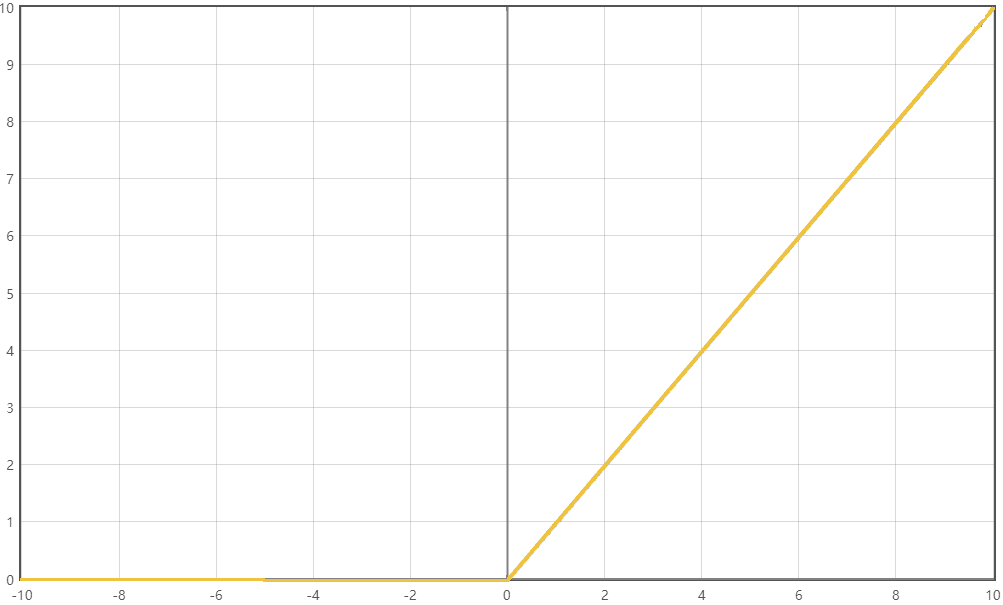


图3.2-5 relu激活函数

在反向传播计算梯度中，使用relu求导明显会比tanh和sigmoid简单，也因此减少了计算量。同时，relu解决了sigomod与tanh引起的梯度消失的问题，使用relu求导时，若输出不为0时，导数均为1，可以有效避免梯度消失问题。另外，relu还会将小于0的映射为0，使得网络较为稀疏，减少神经元之间的依赖，避免过拟合。当然，对应的缺点就是如果学习率使用的太大，很难收敛。

除了这些常用的激活函数外，其实读者也可以自行定义激活函数，激活函数是否可用，只需要满足以下两点。

* 该函数是否单调，只有输出随着输入的增长而增长，在使用梯度下降时寻找局部极值才能成为可能。
* 该函数是否可微，保证该函数定义域内的任意一点上导数都存在，从而使得梯度下降能够正常使用激活函数的输出。

随着大数据时代的到来，硬件设备的快速发展，高效的计算能力解决了训练低效的问题，大量的数据也一定概率的避免了过拟合的发生，深度学习也应运而生。深度学习与神经网络的不同主要在于“深”，其由于隐层数量的增加，模型参数也因此增加，并且深度学习不仅仅只增加了线性情况的神经元，非线性的激活函数也随着隐层的增加而增加，模型能力自然会有提升。如果从另外一个角度看，也可以理解为把输入信号进行逐层的加工，从而把输入信号从与目标没有关联的数据分布转变为与目标联系更加紧密的分布，换言之即把深度学习中浅层的神经网络理解为特征提取器，而后面的层则是对特征的进一步的处理，把浅显的低层特征转变为了高层特征，之后用简单的模型即可完成复杂的机器学习任务。对比传统的机器学习算法来说，通常需要人为的收集整理特征，即特征工程，特征的好坏又对模型结果有巨大的影响并且可能会引入一些人为误差，深度学习也变相解决了特征提取问题，从而成为了一种端到端的模型，这也是为什么近年来深度学习发展如此迅速的其中一个重要原因。