一、深度学习发展史

1. **深度学习的起源阶段**

   1943年，心里学家麦卡洛克和数学逻辑学家皮兹发表论文《神经活动中内在思想的逻辑演算》，提出了MP模型。MP模型是模仿神经元的结构和工作原理，构成出的一个基于神经网络的数学模型，本质上是一种“模拟人类大脑”的神经元模型。MP模型作为人工神经网络的起源，开创了人工神经网络的新时代，也奠定了神经网络模型的基础。

   1949年，加拿大著名心理学家唐纳德·赫布在《行为的组织》中提出了一种基于无监督学习的规则——海布学习规则(Hebb Rule)。海布规则模仿人类认知世界的过程建立一种“网络模型”，该网络模型针对训练集进行大量的训练并提取训练集的统计特征，然后按照样本的相似程度进行分类，把相互之间联系密切的样本分为一类，这样就把样本分成了若干类。海布学习规则与“条件反射”机理一致，为以后的神经网络学习算法奠定了基础，具有重大的历史意义。

   20世纪50年代末，在MP模型和海布学习规则的研究基础上，美国科学家罗森布拉特发现了一种类似于人类学习过程的学习算法——感知机学习。并于1958年，正式提出了由两层神经元组成的神经网络，称之为“感知器”。感知器本质上是一种线性模型，可以对输入的训练集数据进行二分类，且能够在训练集中自动更新权值。感知器的提出吸引了大量科学家对人工神经网络研究的兴趣，对神经网络的发展具有里程碑式的意义。

   但随着研究的深入，在1969年，“AI之父”马文·明斯基和LOGO语言的创始人西蒙·派珀特共同编写了一本书籍《感知器》，在书中他们证明了单层感知器无法解决线性不可分问题（例如：异或问题）。由于这个致命的缺陷以及没有及时推广感知器到多层神经网络中，在20世纪70年代，人工神经网络进入了第一个寒冬期，人们对神经网络的研究也停滞了将近20年。

2. **深度学习的发展阶段**

   1982年，著名物理学家约翰·霍普菲尔德发明了Hopfield神经网络。Hopfield神经网络是一种结合存储系统和二元系统的循环神经网络。Hopfield网络也可以模拟人类的记忆，根据激活函数的选取不同，有连续型和离散型两种类型，分别用于优化计算和联想记忆。但由于容易陷入局部最小值的缺陷，该算法并未在当时引起很大的轰动。

   直到1986年，深度学习之父杰弗里·辛顿提出了一种适用于多层感知器的反向传播算法——BP算法。BP算法在传统神经网络正向传播的基础上，增加了误差的反向传播过程。反向传播过程不断地调整神经元之间的权值和阈值，直到输出的误差达到减小到允许的范围之内，或达到预先设定的训练次数为止。BP算法完美的解决了非线性分类问题，让人工神经网络再次的引起了人们广泛的关注。

   但是由于八十年代计算机的硬件水平有限，如：运算能力跟不上，这就导致当神经网络的规模增大时，再使用BP算法会出现“梯度消失”的问题。这使得BP算法的发展受到了很大的限制。再加上90年代中期，以SVM为代表的其它浅层机器学习算法被提出，并在分类、回归问题上均取得了很好的效果，其原理又明显不同于神经网络模型，所以人工神经网络的发展再次进入了瓶颈期。

3. **深度学习的爆发阶段**

   2006年，杰弗里·辛顿以及他的学生鲁斯兰·萨拉赫丁诺夫正式提出了深度学习的概念。他们在世界顶级学术期刊《科学》发表的一篇文章中详细的给出了“梯度消失”问题的解决方案——通过无监督的学习方法逐层训练算法，再使用有监督的反向传播算法进行调优。该深度学习方法的提出，立即在学术圈引起了巨大的反响，以斯坦福大学、多伦多大学为代表的众多世界知名高校纷纷投入巨大的人力、财力进行深度学习领域的相关研究。而后又在迅速蔓延到工业界中.

   2012年，在著名的ImageNet图像识别大赛中，杰弗里·辛顿领导的小组采用深度学习模型AlexNet一举夺冠。AlexNet采用ReLU激活函数，从根本上解决了梯度消失问题，并采用GPU极大的提高了模型的运算速度。同年，由斯坦福大学著名的吴恩达教授和世界顶尖计算机专家Jeff Dean共同主导的深度神经网络——DNN技术在图像识别领域取得了惊人的成绩，在ImageNet评测中成功的把错

误率从26%降低到了15%。深度学习算法在世界大赛的脱颖而出，也再一次吸引了学术界和工业界对于深度学习领域的关注。

随着深度学习技术的不断进步以及数据处理能力的不断提升，2014年，Facebook基于深度学习技术的DeepFace项目，在人脸识别方面的准确率已经能达到97%以上，跟人类识别的准确率几乎没有差别。这样的结果也再一次证明了深度学习算法在图像识别方面的一骑绝尘。

2016年，随着谷歌公司基于深度学习开发的AlphaGo以4:1的比分战胜了国际顶尖围棋高手李世石，深度学习的热度一时无两。后来，AlphaGo又接连和众多世界级围棋高手过招，均取得了完胜。这也证明了在围棋界，基于深度学习技术的机器人已经超越了人类。

2017年，基于强化学习算法的AlphaGo升级版AlphaGo Zero横空出世。其采用“从零开始”、“无师自通”的学习模式，以100:0的比分轻而易举打败了之前的AlphaGo。除了围棋，它还精通国际象棋等其它棋类游戏，可以说是真正的棋类“天才”。此外在这一年，深度学习的相关算法在医疗、金融、艺术、无人驾驶等多个领域均取得了显著的成果。所以，也有专家把2017年看作是深度学习甚至是人工智能发展最为突飞猛进的一年。

二、人工智能、机器学习、深度学习有什么区别?

1. **人工智能**

   人工智能（Artificial intelligence）简称AI。人工智能是计算机科学的一个分支，它企图了解智能的本质，并生产出一种新的能以人类智能相似的方式做出反应的智能机器，是研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新的技术科学。

2. **机器学习**

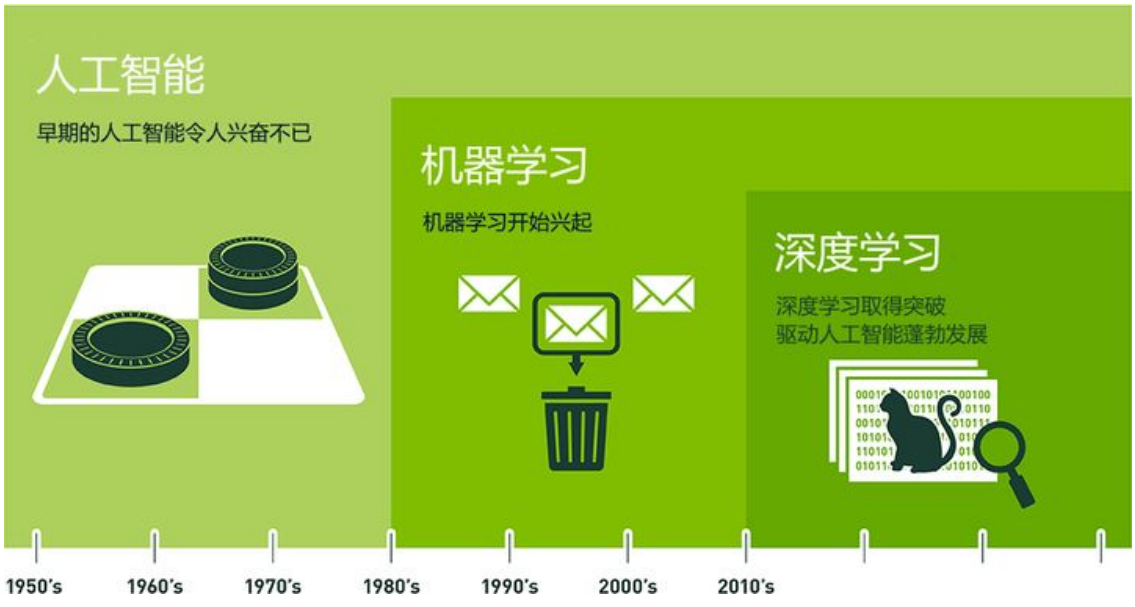
   机器学习（MachineLearning）简称ML。机器学习属于人工智能的一个分支，是实现人工智能的一种方法，也是人工智能的和核心。机器学习理论主要是设计和分析一些让计算机可以自动”学习“的算法。

3. **深度学习**

   深度学习（DeepLearning）简称DL。最初的深度学习是利用深度神经网络来解决特征表达的一种学习过程。深度神经网络本身并不是一个全新的概念，可大致理解为包含多个隐含层的神经网络结构。为了提高深层神经网络的训练效果，人们对神经元的连接方法和激活函数等方面做出相应的调整。深度学习是机器学习研究中的一个新的领域，其动机在于建立、模拟人脑进行分析学习的神经网络，它模仿人脑的机制来解释数据，如图象、声音、文本。

4. **三者的区别与联系**

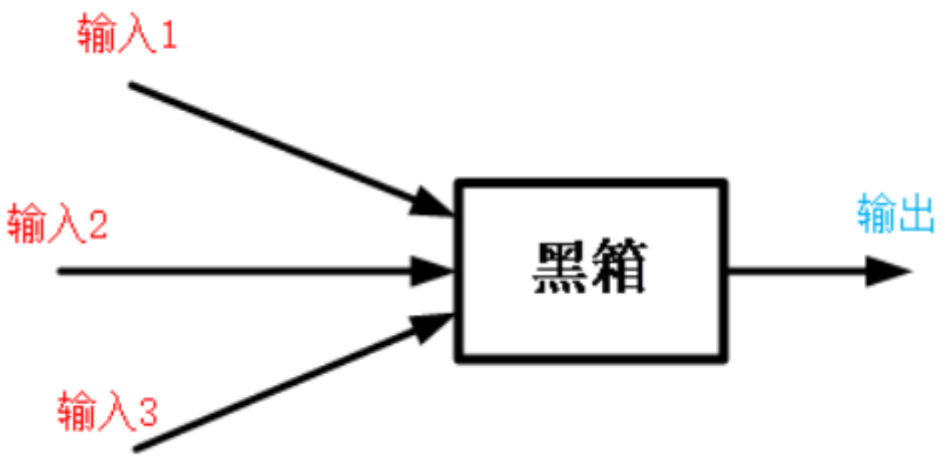
   三者类似于一个包含关系，具体可以用下面这张图来说明：



三、神经元、单层感知机、多层感知机

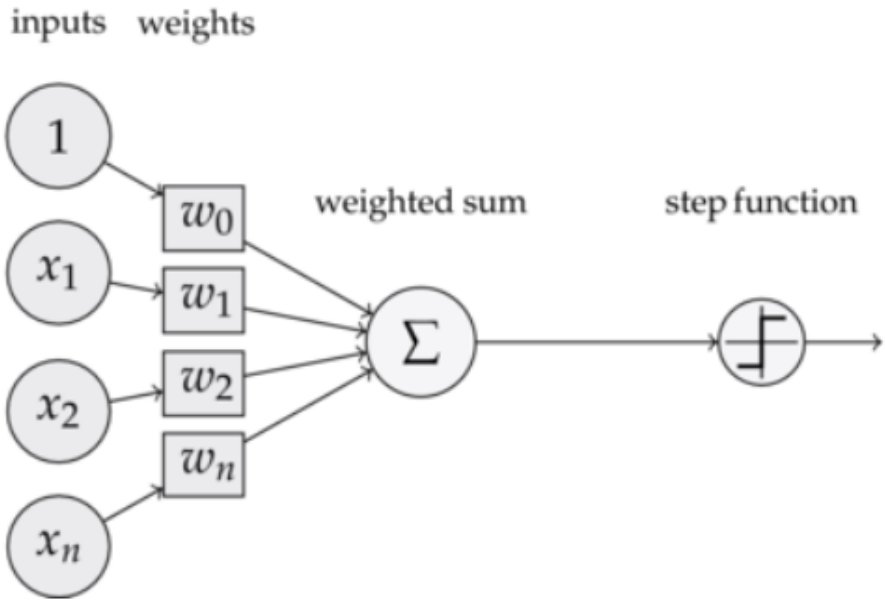
1. **神经元**

   一个神经元通常具有多个树突，主要用来接受传入信息；而轴突只有一条，轴突尾端有许多轴突末梢可以给其他多个神经元传递信息。轴突末梢跟其他神经元的树突产生连接，从而传递信号。这个连接的位置在生物学上叫做“突触”。突触之间的交流通过神经递质实现。它的数学模型如下：



2. **单层感知机**

   将神经元模型中的黑箱进一步细化以及符号化就可以得到单层感知机模型，示意图如下：



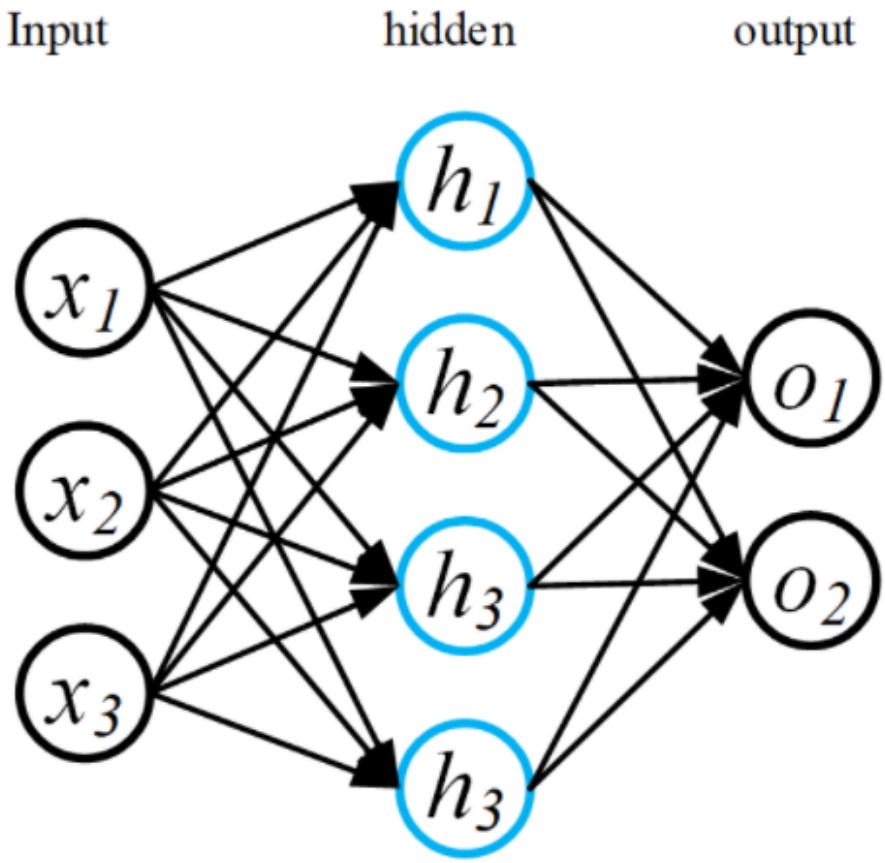
   可以看到，感知机的基本模型包括：

   - **输入**：$x_1, x_2, \ldots, x_n$，实际可能回比这更多,此处添加了一个偏置1，是为了平衡线性加权函数总是过零点的问题。
   - **权值**：对应于每个输入都有一个加权的权值$w_1, w_2, \ldots, w_n$
   - **激活函数**：激活函数f对应于一个非线性函数，其选择有很多，本文后面会详细介绍
   - **输出y**：由激活激活函数进行处理后的结果，往往是区分度较大的非连续值用于分类。
     $y = f(w_0 + x_1 * w_1 + x_2 * w_2 + \cdots + x_n * w_n)$

   单层感知机缺点是无法解决“异或”问题

3. **多层感知机**

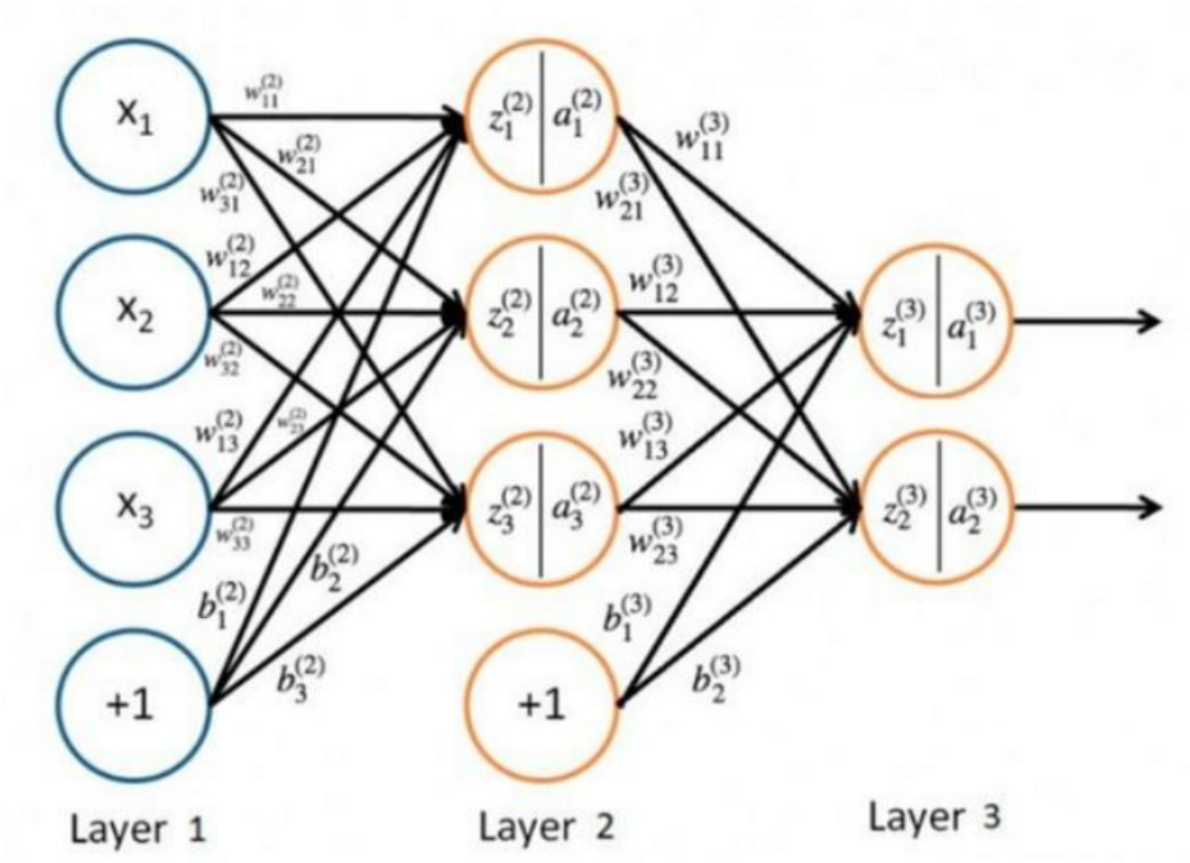
为了弥补单层感知机无法解决“异或”问题的缺陷，多层感知机出现了，多层感知机就是含有至少一个隐藏层的由全连接层组成的神经网络，且在整个神经网络中除了输入层之外，每一层都经过激活函数进行非线性变换。多层感知机的层数和各隐藏层中隐藏单元个数都是超参数。参考下图的单隐藏层的MLP结构：



多层感知机采用bp算法进行优化，利用一次前向过程计算得到的误差，反向传播到每一层的权重上，并实现权重的逐步更新，通过足够多的更新步骤后得到一个较优权重使最终得到的误差能够满足运算标准。

四、什么是前向传播（包含图文示例）

所谓的前向传播算法就是：**将上一层的输出作为下一层的输入，并计算下一层的输出，一直到运算到输出层为止**



对于Layer 2的输出 $a_1^{(2)}$ ， $a_2^{(2)}$， $a_3^{(2)}$，

$$a_1^{(2)} = \sigma(z_1^{(2)}) = \sigma(w_{11}^{(2)} x_1 + w_{12}^{(2)} x_2 + w_{13}^{(2)} x_3 + b_1^{(2)})$$
$$a_2^{(2)} = \sigma(z_2^{(2)}) = \sigma(w_{21}^{(2)} x_1 + w_{22}^{(2)} x_2 + w_{23}^{(2)} x_3 + b_2^{(2)})$$
$$a_3^{(2)} = \sigma(z_3^{(2)}) = \sigma(w_{31}^{(2)} x_1 + w_{32}^{(2)} x_2 + w_{33}^{(2)} x_3 + b_3^{(2)})$$

对于Layer 3的输出$a_1^{(3)}$，

$$a_1^{(3)} = \sigma(z_1^{(3)}) = \sigma(w_{11}^{(3)} a_1^{(2)} + w_{12}^{(3)} a_2^{(2)} + w_{13}^{(3)} a_3^{(2)} + b_1^{(3)})$$
$$a_2^{(3)} = \sigma(z_2^{(3)}) = \sigma(w_{21}^{(3)} a_1^{(2)} + w_{22}^{(3)} a_2^{(2)} + w_{23}^{(3)} a_3^{(2)} + b_2^{(3)})$$

写成矩阵乘法的形式：

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$
$$a^{(l)} = \sigma(z^{(l)})$$

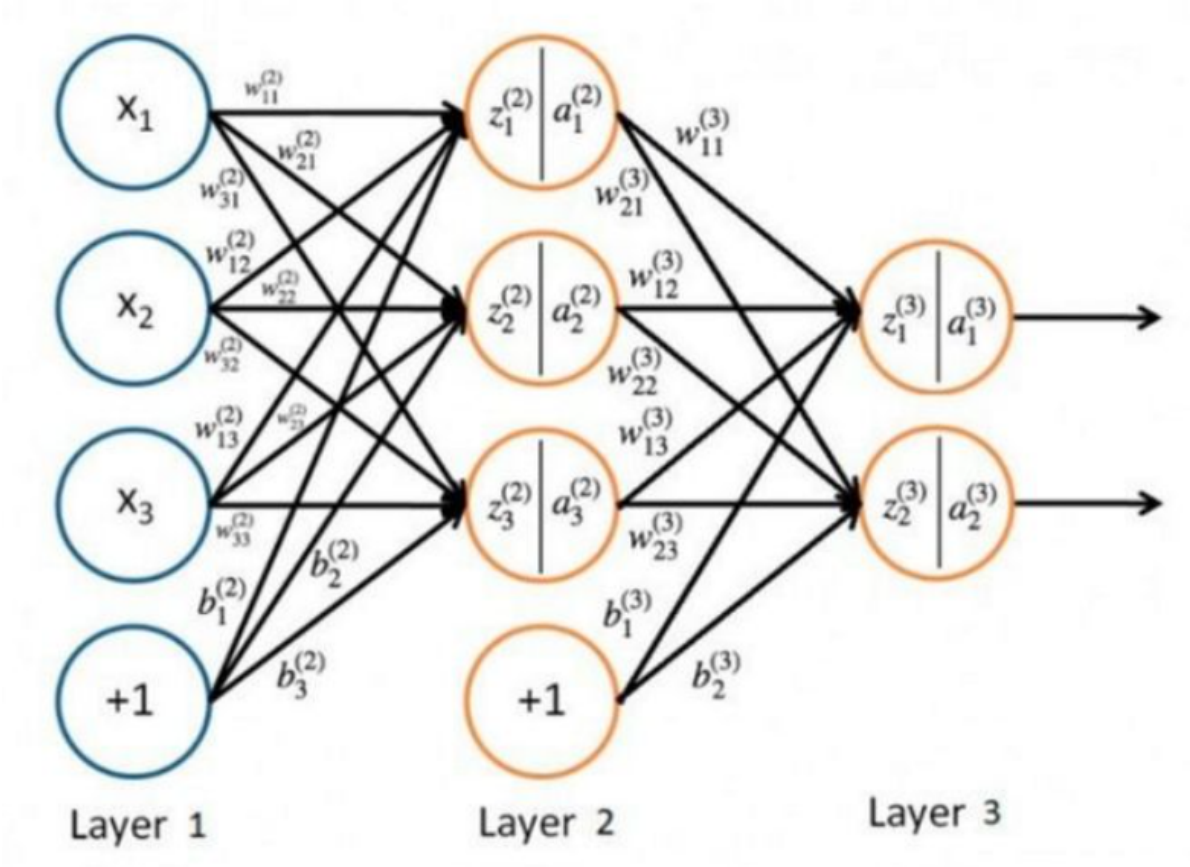
其中 σ 为 sigmoid 函数

五、什么是反向传播（包含图文示例）

反向传播（back propagation, **BP**）算法是 "误差反向传播" 的简称，也称为**backprop**，允许来自代价函数的信息通过网络向后流动，以便计算梯度。

微积分中的链式法则（为了不与概率中的链式法则相混淆）用于计复合函数的导数。反向传播是一种计算链式法则的算法，使用高效的特定运输顺序。

设 x 是实数， f 和 g 是从实数映射到实数的函数。假设 $y = g(x)$ 并且 $z = f(g(x)) = f(y)$。那么链式法则就是： $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$。



在进行反向传播算法前，我们需要选择一个损失函数，来度量训练样本计算出的输出和真实的训练样本输出之间的损失。我们使用最常见的**均方误差（MSE）来作为损失函数，**

$$C(W,b) = \frac{1}{2}\left\|a^{(l)} - y\right\|_2^2$$

其中 $a^{(l)}$ 为训练样本计算出的输出，y为训练样本的真实值。加入系数 $\frac{1}{2}$ 是为了抵消微分出来的指数。

(1) 输出层的梯度

$$\begin{aligned}\frac{\partial C(W,b)}{\partial w^{(l)}} &= \frac{\partial C(W,b)}{\partial a^{(l)}}\frac{\partial a^{(l)}}{\partial z^{(l)}}\frac{\partial z^{(l)}}{\partial w^{(l)}} \\ &= \left(a^{(l)} - y\right) \odot \sigma'\left(z^{(l)}\right) a^{(l-1)} \\ \frac{\partial C(W,b)}{\partial b^{(l)}} &= \frac{\partial C(W,b)}{\partial a^{(l)}}\frac{\partial a^{(l)}}{\partial z^{(l)}}\frac{\partial z^{(l)}}{\partial b^{(l)}} \\ &= \left(a^{(l)} - y\right) \odot \sigma'\left(z^{(l)}\right)\end{aligned}$$

其中 $\odot$ 表示Hadamard积，即两个维度相同的矩阵对应元素的乘积。

我们注意到在求解输出层梯度的时候有公共的部分，记为

$$\delta^{(l)} = \frac{\partial C(W,b)}{\partial a^{(l)}}\frac{\partial a^{(l)}}{\partial z^{(l)}} = \left(a^{(l)} - y\right) \odot \sigma'\left(z^{(l)}\right)$$

(2) 隐藏层的梯度

我们把输出层 l 的梯度算出来了，那么如何计算 $l-1$ 层的梯度，$l-2$ 层的梯度呢?

因为上面已经求出了输出层的误差，根据误差反向传播的原理，当前层的误差可理解为上一层所有神经元误差的复合函数，即使用上一层的误差来表示当前层误差，并依次递推。

这里我们用数学归纳法，假设第 $l+1$ 层的 $\delta^{(l+1)}$ 已经求出，那么我们如何求出第 l 层的$\delta^{(l)}$ 呢?

$$\delta^{(l)}=\frac{\partial C(W,b)}{\partial z^{(l+1)}}\frac{\partial z^{(l+1)}}{\partial z^{(l)}}=\delta^{(l+1)}\frac{\partial z^{(l+1)}}{\partial z^{(l)}}$$

而$z^{(l+1)}$和$z^{(l)}$的关系如下:

$$z^{(l+1)}=W^{(l+1)}a^{(l)}+b^{(l+1)}=W^{(l+1)}\sigma\left(z^{(l)}\right)+b^{(l+1)}$$

这样很容易求出，

$$\frac{\partial z^{(l+1)}}{\partial z^{(l)}}=\left(W^{(l+1)}\right)^{T}\odot\sigma'\left(z^{(l)}\right)$$

所以，

$$\delta^{(l)}=\left(W^{(l+1)}\right)^{T}\delta^{(l+1)}\odot\sigma'\left(z^{(l)}\right)$$

现在我们得到了 $\delta^{(l)}$ 的递推关系式，只要求出了某一层的$\delta^{(l)}$，求解$w^{(l)}$，$b^{(l)}$的对应梯度就很简单:

$$\frac{\partial C(W,b)}{\partial w^{(l)}}=\delta^{(l)}a^{(l-1)}$$
$$\frac{\partial C(W,b)}{\partial b^{(l)}}=\delta^{(l)}$$

总结一下反向传播的流程:

输入: 总层数L，以及各隐藏层与输出层的神经元个数，激活函数，损失函数，迭代步长 α，最大迭代次数MAX与停止迭代阈值 ϵ，输入的m个训练样本 $((x_1,y_1),(x_2,y_2),...(x_m,y_m))$

1. 初始化参数W，b

2. 进行前向传播算法计算，for $l=2$ to L

$$z^{(l)}=W^{(l)}a^{(l-1)}+b^{(l)}$$
$$a^{(l)}=\sigma\left(z^{(l)}\right)$$

3. 通过损失函数计算输出层的梯度

4. 进行反向传播算法计算，for $l=L-1$ to 2

$$\delta^{(l)}=\left(W^{(l+1)}\right)^{T}\delta^{(l+1)}\odot\sigma'\left(z^{(l)}\right)$$

5. 更新W，b

通过梯度下降算法更新权重w和偏置b的值，α为学习率其中$\alpha\in(0,1]$。

$$w^{(l)}=w^{(l)}-\alpha\frac{\partial C(w,b)}{\partial w^{(l)}}$$

$$b^{(l)}=b^{(l)}-\alpha\frac{\partial C(w,b)}{\partial b^{(l)}}$$

6. 如果所有W，b的变化值都小于停止迭代阈值ϵ，则跳出迭代循环

7. 输出各隐藏层与输出层的线性关系系数矩阵W和偏置b。

Reference

[1] 人工智能、机器学习和深度学习有什么区别？ - 知乎 (zhihu.com)

[2] 人工智能发展简史 - AMiner

[3] (20条消息) 1、反向传播——通俗易懂_u013049912的博客-CSDN博客