西安电子科技大学

题　　目：PaddlePaddle 作业

学　　院：人工智能学院

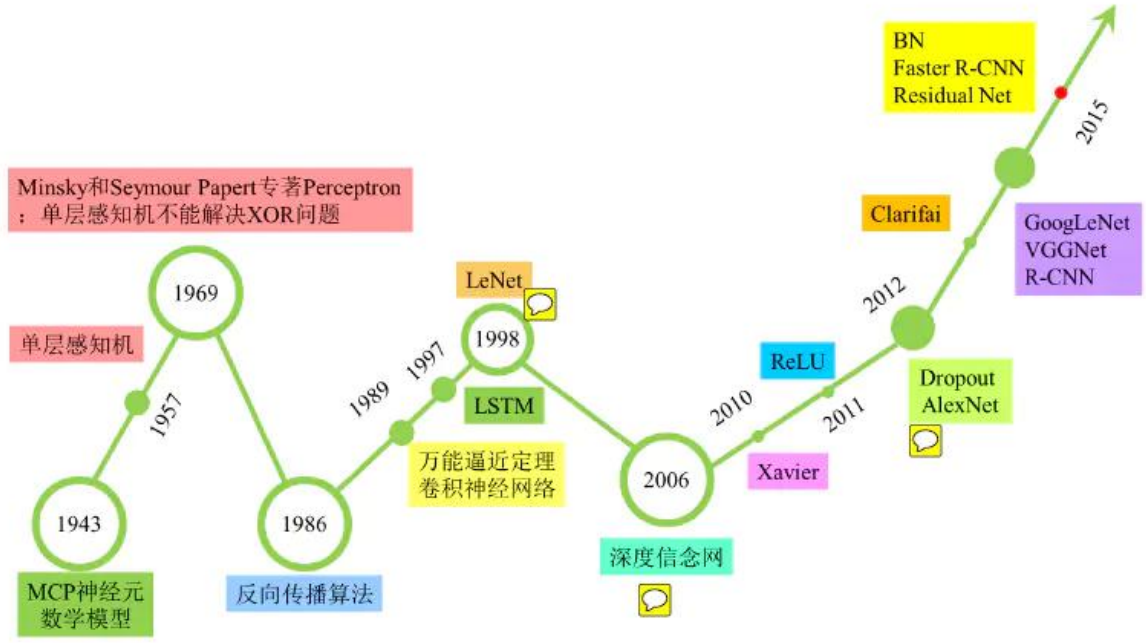
专　　业：智能科学与技术

学生姓名：秦晨浩

学　　号：18200100169

1、深度学习发展历史：

一般来说，目前为止已经有三次深度学习的发展浪潮：在 20 世纪 40 年代到 60 年代深度学习被称为控制论，20 世纪 80 年代到 90 年代深度学习被誉为连接机制 ，并于 2006 年开始，以深度学习之名复兴。总体发展史如下：



由图可以明显看出 DL 在从 06 年崛起之前经历了两个低谷，这两个低谷也将神经网络的发展分为了三个不同的阶段。

第一代神经网络（1958~1969）

最早的神经网络的思想起源于 1943 年的 MCP 人工神经元模型，当时是希望能够用计算机来模拟人的神经元反应的过程，该模型将神经元简化为了三个过程：输入信号线性加权，求和、非线性激活（阈值法）。

第一次将 MCP 用于机器学习（分类）的当属 1958 年 Rosenblatt 发明的感知器（perceptron）算法。该算法使用 MCP 模型对输入的多维数据进行二分类，且能够使用梯度下降法从训练样本中自动学习更新权值。1962 年，该方法被证明为能够收敛，理论与实践效果引起第一次神经网络的浪潮。

然而学科发展的历史不总是一帆风顺的。

1969 年，美国数学家及人工智能先驱 Minsky 在其著作中证明了感知器本质上是一种线性模型，只能处理线性分类问题，就连最简单的 XOR（亦或）问题都无法正确分类。这等于直接宣判了感知器的死刑，神经网络的研究也陷入了近 20 年的停滞。

第二代神经网络（1986~1998）

第一次打破非线性诅咒的当属现代 DL 大牛 Hinton，其在 1986 年发明了适用于多层感知器（MLP）的 BP 算法，并采用 Sigmoid 进行非线性映射，有效解决了非线性分类和学习的问题。该方法引起了神经网络的第二次热潮。

1989 年，Robert Hecht-Nielsen 证明了 MLP 的万能逼近定理，即对于任何闭区间内的一个连续函数 f，都可以用含有一个隐含层的 BP 网络来逼近该定理的发现极大的鼓舞了神经网络的研究人员。

同年，LeCun 发明了卷积神经网络-LeNet，并将其用于数字识别，且取得了较好的成绩，不过当时并没有引起足够的注意。

值得强调的是在1989年以后由于没有特别突出的方法被提出，且NN一直缺少相应的严格的数学理论支持，神经网络的热潮渐渐冷淡下去。冰点来自于1991年，BP算法被指出存在梯度消失问题，即在误差梯度后向传递的过程中，后层梯度以乘性方式叠加到前层，由于Sigmoid函数的饱和特性，后层梯度本来就小，误差梯度传到前层时几乎为0，因此无法对前层进行有效的学习，该发现对此时的NN发展雪上加霜。

1997年，LSTM模型被发明，尽管该模型在序列建模上的特性非常突出，但由于正处于NN的下坡期，也没有引起足够的重视。

统计学习方法（1986~2006）

1986年，决策树方法被提出，很快ID3，ID4，CART等改进的决策树方法相继出现，到目前仍然是非常常用的一种机器学习方法。该方法也是符号学习方法的代表。

1995年，线性SVM被统计学家Vapnik提出。该方法的特点有两个：由非常完美的数学理论推导而来（统计学与凸优化等），符合人的直观感受（最大间隔）。不过，最重要的还是该方法在线性分类的问题上取得了当时最好的成绩。

1997年，AdaBoost被提出，该方法是PAC（Probably Approximately Correct）理论在机器学习实践上的代表，也催生了集成方法这一类。该方法通过一系列的弱分类器集成，达到强分类器的效果。

2000年，KernelSVM被提出，核化的SVM通过一种巧妙的方式将原空间线性不可分的问题，通过Kernel映射成高维空间的线性可分问题，成功解决了非线性分类的问题，且分类效果非常好。至此也更加终结了NN时代。

2001年，随机森林被提出，这是集成方法的另一代表，该方法的理论扎实，比AdaBoost更好的抑制过拟合问题，实际效果也非常不错。

2001年，一种新的统一框架-图模型被提出，该方法试图统一机器学习混乱的方法，如朴素贝叶斯，SVM，隐马尔可夫模型等，为各种学习方法提供一个统一的描述框架。

第三代神经网络-DL（2006-至今）

该阶段又分为两个时期：快速发展期（2006~2012）与爆发期（2012~至今）

快速发展期（2006~2012）

2006年，DL元年。是年，Hinton提出了深层网络训练中梯度消失问题的解决方案：无监督预训练对权值进行初始化+有监督训练微调。其主要思想是先通过自学习的方法学习到训练数据的结构（自动编码器），然后在该结构上进行有监督训练微调。但是由于没有特别有效的实验验证，该论文并没有引起重视。

2011年，ReLU激活函数被提出，该激活函数能够有效的抑制梯度消失问题。

2011年，微软首次将DL应用在语音识别上，取得了重大突破。

爆发期（2012~至今）

2012年，Hinton课题组为了证明深度学习的潜力，首次参加ImageNet图像识别比赛，其通过构建的CNN网络AlexNet一举夺得冠军，且碾压第二名（SVM方法）的分类性能。也正是由于该比赛，CNN吸引到了众多研究者的注意。

2013,2014,2015年，通过ImageNet图像识别比赛，DL的网络结构，训练方法，GPU硬件的不断进步，促使其在其他领域也在不断的征服战场

2015年，Hinton，LeCun，Bengio论证了局部极值问题对于DL的影响，结果是Loss的局部极值问题对于深层网络来说影响可以忽略。该论断也消除了笼罩在神经网络上的局部极值问题的阴霾。具体原因是深层网络虽然局部极值非常

多，但是通过 DL 的 BatchGradientDescent 优化方法很难陷进去，而且就算陷进去，其局部极小值点与全局极小值点也是非常接近，但是浅层网络却不然，其拥有较少的局部极小值点，但是却很容易陷进去，且这些局部极小值点与全局极小值点相差较大。

2015，DeepResidualNet 发明。分层预训练，ReLU 和 BatchNormalization 都是为了解决深度神经网络优化时的梯度消失或者爆炸问题。但是在对更深层的神经网络进行优化时，又出现了新的 Degradation 问题，即”通常来说，如果在 VGG16 后面加上若干个单位映射，网络的输出特性将和 VGG16 一样，这说明更深次的网络其潜在的分类性能只可能>=VGG16 的性能，不可能变坏，然而实际效果却是只是简单的加深 VGG16 的话，分类性能会下降（不考虑模型过拟合问题）“Residual 网络认为这说明 DL 网络在学习单位映射方面有困难，因此设计了一个对于单位映射（或接近单位映射）有较强学习能力的 DL 网络，极大的增强了 DL 网络的表达能力。此方法能够轻松的训练高达 150 层的网络。

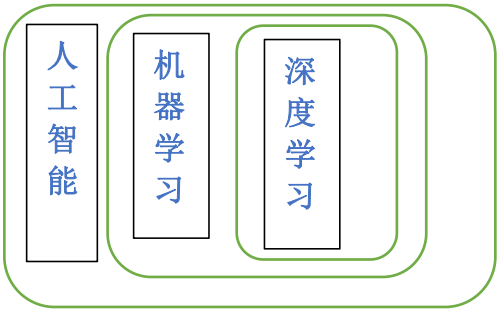
2、人工智能、机器学习、深度学习有什么联系与区别？

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。

机器学习属于人工智能的一个分支，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。

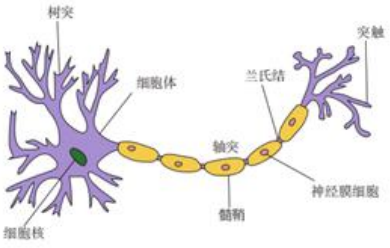
深度学习是利用深度神经网络来解决特征表达的一种学习过程。其动机在于建立、模拟人脑进行分析学习的神经网络，它模仿人脑的机制来解释数据，如图象、声音、文本。

总结来说，机器学习是一种实现人工智能的方法，而深度学习是机器学习中的一种技术，三者关系可以如下表示：



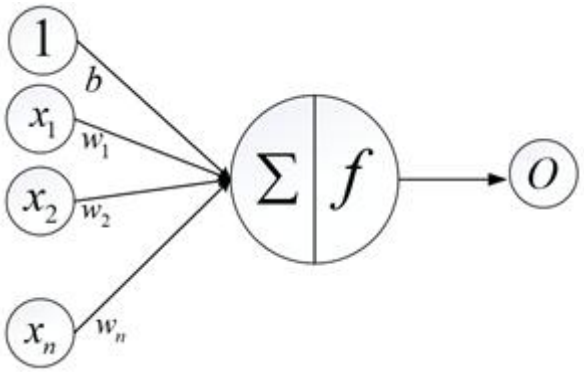
3、神经元、单层感知机、多层感知机：

大脑神经系统是由成千上万的生物神经元通过突触相互连接的复杂神经网络，因此神经元是神经系统的基本组成单元，用于产生神经冲动和传输神经冲动。下图展示了一个典型的生物神经元结构：

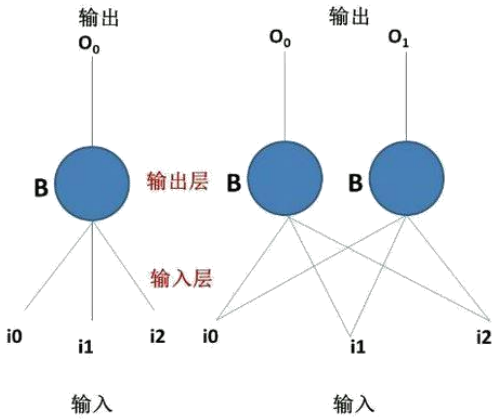


生物神经元主要由细胞体、树突和轴突组成。
（1）树突：从细胞体向四周分散开来的突起称之为树突，用于接收神经冲动，神经元细胞有一个或多个树突。与不同神经元连接的树突强度有强有弱；
（2）细胞体：细胞体细胞膜上存在受体，可以与神经冲动的神经递质结合，产生兴奋或者抑制。神经元所获得的输入信号的积累超过某个阈值时，处于兴奋状态，否则为抑制状态；
（3）轴突：将自身的兴奋传递到下一个神经元细胞的树突

人工神经元是人工神经网络的基本单元。模拟生物神经元，人工神经元有 1 个或者多个输入（模拟多个树突或者多个神经元向该神经元传递神经冲动）；对输入进行加权求和（模拟细胞体将神经信号进行积累和树突强度不同）；对输入之和使用激活函数计算活性值（模拟细胞体产生兴奋或者抑制）；输出活性值并传递到下一个人工神经元（模拟生物神经元通过轴突将神经冲动输入到下一个神经元）。

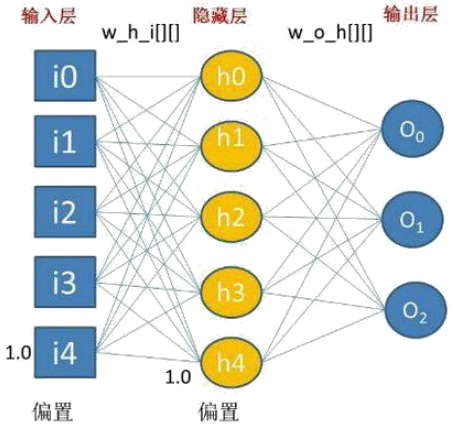


单层感知器可以用来区分线性可分的数据，并且一定可以在有限的迭代次数中收敛。 单层感知器能够用来模拟逻辑函数，例如逻辑非 NOT、逻辑或非 XOR、逻辑或 OR、逻辑与 AND 和逻辑与非 NAND 等，但是不能用来模拟逻辑异或 XOR（对于这个函数，必须用两层神经元）



单层感知器有一个缺点，即它们只能对线性可分的数据集进行分类。然而，此后不久人们就发现，如果见单层感知器叠加成多层感知器，则理论上就具备对任何分类问题的解决能力。事实上，多层感知器可以模拟成任意复杂的函数，其中函数的复杂性取决于网络输入个数和隐层个数。

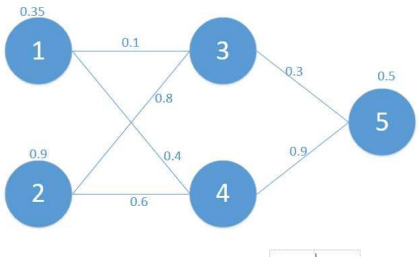
多层感知器中神经元的性质和单层感知器中的一样（例如偏置等）。然而由于存在多层，多层感知器中某一层的输出就是下一层的输入。因此多层感知器的实现比单层感知器略显复杂，但还是很直观的。注意，在这里对隐层结点和输出层结点使用相同的激励函数，即 sigmoid 函数。Sigmoid 函数可以将神经元的输出值规格化到区间[0,1]。



4、什么是前向传播？

通过输入层输入，一路向前，通过输出层输出的一个结果。如图指的是 1 、x1、x2、xn、与权重（weights）相乘，并且加上偏置值 b0，然后进行总的求和，同时通过激活函数激活之后算出结果。这个过程就是前向传播。

如图：



$$X = Z_0 = \begin{bmatrix} 0.35 \\ 0.9 \end{bmatrix}$$

$$y_{out} = 0.5$$

$$W0 = \begin{bmatrix} w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.8 \\ 0.4 & 0.6 \end{bmatrix}$$

$$W1 = \begin{bmatrix} w_{53} & w_{54} \end{bmatrix} = \begin{bmatrix} 0.3 & 0.9 \end{bmatrix}$$

那么：

$$z1 = \begin{bmatrix} z_3 \\ z_4 \end{bmatrix} = w0 * X = \begin{bmatrix} w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= \begin{bmatrix} w_{31} * x_1 + w_{32} * x_2 \\ w_{41} * x_1 + w_{42} * x_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 * 0.35 + 0.8 * 0.9 \\ 0.4 * 0.35 + 0.6 * 0.9 \end{bmatrix}$$

$$= \begin{bmatrix} 0.755 \\ 0.68 \end{bmatrix}$$

$$y1 = \begin{bmatrix} y_3 \\ y_4 \end{bmatrix} = f(w0 * X) = f\left(\begin{bmatrix} w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)$$

$$= f\left(\begin{bmatrix} w_{31} * x_1 + w_{32} * x_2 \\ w_{41} * x_1 + w_{42} * x_2 \end{bmatrix}\right)$$

$$= f\left(\begin{bmatrix} 0.755 \\ 0.68 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 0.680 \\ 0.663 \end{bmatrix}$$

$$z2 = w1 * y1 = \begin{bmatrix} w_{53} & w_{54} \end{bmatrix} * \begin{bmatrix} y_3 \\ y_4 \end{bmatrix}$$

$$= \begin{bmatrix} w_{53} * y_3 + w_{54} * y_4 \end{bmatrix}$$

$$= \begin{bmatrix} 0.801 \end{bmatrix}$$

$$y2 = f(z2) = f(w1 * y1) = f\left(\begin{bmatrix} w_{53} & w_{54} \end{bmatrix} * \begin{bmatrix} y_3 \\ y_4 \end{bmatrix}\right)$$

则

$$= f\left(\begin{bmatrix} w_{53} * y_3 + w_{54} * y_4 \end{bmatrix}\right)$$

$$= f\left(\begin{bmatrix} 0.801 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 0.690 \end{bmatrix}$$

则损失 C：$C=\frac{1}{2}(0.690-0.5)^2=0.01805$

5、什么是反向传播？

通过输出反向更新权重的过程。具体的说输出位置会产生一个模型的输出，通过这个输出以及原数据计算一个差值。将前向计算过程反过来计算。通过差值和学习率更新权重。同上例：

$$\begin{cases} C=\frac{1}{2}(y_2-y_{out})^2 \\ y_2=f(z_2) \\ z_2=(w_{53}*y_3+w_{54}*y_4) \end{cases}$$

求出 C 对 W 偏导，链式法则：

$$\begin{aligned} &\frac{\partial C}{\partial w_{53}}=\frac{\partial C}{\partial y_5}*\frac{\partial y_5}{\partial z_5}*\frac{\partial z_5}{\partial w_{53}} \\ &=(y_5-y_{out})*f(z_2)*(1-f(z_2))*y_3 \\ &=(0.69-0.5)*(0.69)*(1-0.69)*0.68 \\ &=0.02763 \end{aligned}$$

$$\begin{aligned} &\frac{\partial C}{\partial w_{54}}=\frac{\partial C}{\partial y_5}*\frac{\partial y_5}{\partial z_5}*\frac{\partial z_5}{\partial w_{53}} \\ &=(y_5-y_{out})*f(z_5)*(1-f(z_5))*y_4 \\ &=(0.69-0.5)*(0.69)*(1-0.69)*0.663 \\ &=0.02711 \end{aligned}$$

同理可得：

$$\begin{cases} w_{31}=w_{31}-\frac{\partial C}{\partial w_{31}}=0.09661944 \\ w_{32}=w_{32}-\frac{\partial C}{\partial w_{32}}=0.78985831 \\ w_{41}=w_{41}-\frac{\partial C}{\partial w_{41}}=0.39661944 \\ w_{42}=w_{42}-\frac{\partial C}{\partial w_{42}}=0.58985831 \end{cases}$$

则一次反向传播完毕。