1.深度学习发展历史

1.深度学习分为三个时期：

- `1940s-1960s`：这时它被称作控制论`cybernetics`。

- `1980s-1990s`：这时它被称作连接机制`connectionism`。

- `2006--`：这时被称作`deep learning`。

深度学习在历史上的不同名字反映了不同的哲学观点。

2.下图展示了神经网络研究的三个历史浪潮中的两个（因为第三波太近了）。

-第一波随着生物学习理论的发展和第一个模型的出现（如感知机神经元）。

-第二波用反向传播训练一层或者两层隐层神经网络。

3.第一波浪潮：早期的人工智能算法模拟生物的学习过程：对大脑的学习过程建模。

-此时的深度学习被称作人工神经网络`artiﬁcial neural networks:ANNs`，深度学习模型因此被认为是受生物大脑启发的工程系统。

-现代的深度学习超越了神经科学的观点：它是一种多层次学习的、通用的机器学习框架，而不必是从神经科学中获取灵感。

4.第二波浪潮：连接机制的中心思想是：大量简单的计算单元在连接时可以实现智能行为。

-这种观点适用于生物神经系统中的神经元，以及深度网络模型中的隐层神经元。

-在连接机制期间，有一些核心思想仍然影响了后续的神经网络：

-分布式表达`distributed representation`：系统的每个输入应该由许多特征表示，每个特征描述了输入的一个部分。

如一个视觉识别系统可以识别：汽车、卡车、鸟，这些对象可以为红色、蓝色、绿色。

-表达这些输入的一种方式为：9个神经元分别表示红色卡车、红色汽车、红色鸟、绿色汽车...等等。

-如果使用分布式表达，则使用6个神经元：3个神经元来表示卡车、汽车、鸟，另外3个神经元来表示红色、绿色、蓝色。

-反向传播算法`back-propagation`：它是当前主要的训练深度模型的算法。

- `long short-term memory:LSTM`网络：它是一种序列模型，解决了许多自然语言处理任务。

5.第三波浪潮从2006年开始突破。

`Geoﬀrey Hinton`给出了一种称作深度信念网络（`deep belief network`），该网络可以使用`greedy layer-wise pre-training`策略来有效地训练。

# 2.人工智能、机器学习、深度学习有什么区别和联系？

# 3.神经元、单层感知机、多层感知机

1.神经元

归纳为：

l 感觉神经元（传入神经元）：其树突的末端分布于身体的外周部，接收来自体内外的刺激，将兴奋传至脊髓和脑。

l 运动神经元（传出神经元）：其轴突大于肌肉和腺体。

l 联络神经元（中间神经元）：结余上述2种神经元之间，起着神经元之间技能联系的作用，多存在于脑和脊髓里。

2.感知器

感知器（Perceptron），是神经网络中的一个概念，在1950s由Frank Rosenblatt第一次引入。

2.1单层感知器

单层感知器（Single Layer Perceptron）是最简单的神经网络。它包含输入层和输出层，而输入层和输出层是直接相连的。

2.2多层感知器

多层感知器（Multi-Layer Perceptrons），包含多层计算。相对于单层感知器，输出端从一个变到了多个；输入端和输出端之间也不光只有一层。

# 4.什么是前向传播

前向传播

对于一个还没有训练好的神经网络而言，各个神经元之间的参数都是随机值，即初始化时赋的值，前向传播过程是神经网络的输入输出过程，即网络是如何根据X的值得到输出的Y值的。

设激活函数为f，权重矩阵为W，偏置项为b，输入为A，输出为Y，则Y = f(AW+b)，计算Y这个过程就是前向传播的过程。

注意：权重矩阵W一开始是个随机值

在样本中我们的数据是(x,label)，而y是我们经过神经网络计算出来的值，也就是估计值，y和label肯定有误差，如何减少误差，需要改变权重矩阵W，反向传播就是解决这个问题的。

# 5.什么是反向传播

反向传播和梯度下降

反向传播是为了减小误差，使得误差最小的权重矩阵的值便是神经网络最终的权重矩阵。对于误差的描述可以采用代价（误差）函数来描述，比如高中学的:，其中y(x)是神经网络的预测值，label(x)是x对应的真实值，可以看出loss是关于权重矩阵的多元函数，当loss最小时W的取值，便是神经网络权重矩阵最终值。

反向传播的核心算法是梯度下降算法，既然反向传播是为了使loss最小，使得loss收敛的最快。根据场论的知识，梯度的方向是函数值增长最快的方向，那么梯度的反方向便是函数值减少最快的方向。基于这一理论，便有了梯度下降法。

梯度下降法的基本思想可以类比为一个下山的过程。假设这样一个场景：一个人被困在山上，需要从山上下来(找到山的最低点，也就是山谷)。但此时山上的浓雾很大，导致可视度很低。因此，下山的路径就无法确定，他必须利用自己周围的信息去找到下山的路径。这个时候，他就可以利用梯度下降算法来帮助自己下山。具体来说就是，以他当前的所处的位置为基准，寻找这个位置最陡峭的地方，然后朝着山的高度下降的地方走，同理，如果我们的目标是上山，也就是爬到山顶，那么此时应该是朝着最陡峭的方向往上走。然后每走一段距离，都反复采用同一个方法，最后就能成功的抵达山谷。