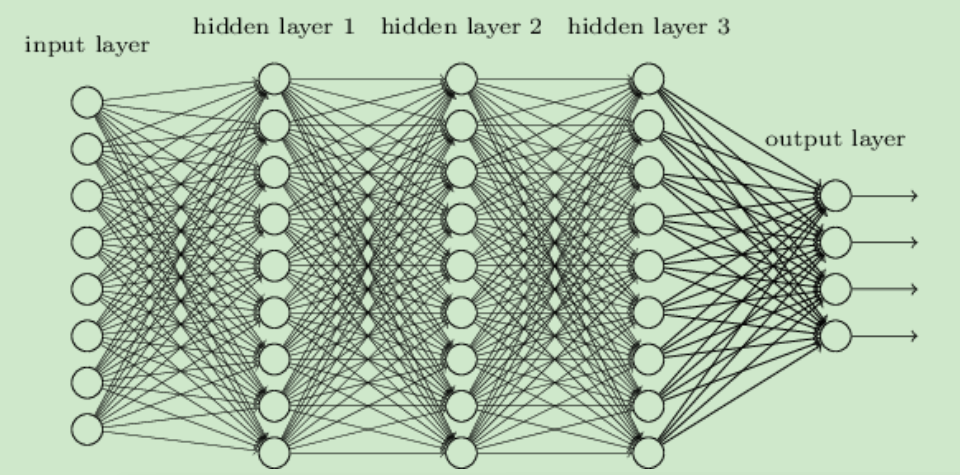
神经网络

**1.神经网络(NN)基本结构**



神经网络第一层为输入层，最后一层为输出层，中间的层都为隐藏层。

层与层之间是全连接的，也就是说，第i层的任意一个神经元与第i+1层的任意一个神经元相连。

已知，第层的输出即第层的输入，第层的权重矩阵，偏移向量，则第层的输出为，

**2.前向传播算法**

所谓的前向传播算法就是利用我们的若干个权重系数矩阵,偏倚向量来和输入值向量进行一系列线性运算和激活运算，从输入层开始，一层层的向后计算，一直到运算到输出层，得到输出结果为值。

已知，第层的输出即第层的输入，第层的权重矩阵，偏移向量，则第层的输出为

其中，为激活函数，常见的激活函数有sigmoid、relu等。

假设第层共有*m*个神经元，第层共有*n*个神经元，则矩阵的形状为，第层的偏移为的向量，第层的输出的形状为,第层激活前的输出为的向量，第层的输出为

的向量。

流程：

输入：总层数L，所有隐藏层和输出层对应的矩阵,偏倚向量，输入值向量

输出：输出层的输出

1. 初始化

（2） ,计算：

最后的结果即为输出

**3.反向传播算法**

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。

对于回归问题，常见的损失函数为均方误差。

和为特征维度为的向量,而为S的L2范数。

损失函数有了，现在我们开始用梯度下降法迭代求解每一层的,。

首先是输出层第*L*层。输出层的,满足下式：

对于输出层的参数，损失函数变为：

=

注意上式中有一个符号,它代表Hadamard积，对于两个维度相同的向量和,则 ,即对应位置的元素相乘。

记

对于第层的未激活输出，他的梯度可以表示为：

如果我们可以依次计算出第层的,则该层的,很容易计算，注意到根据前向传播算法，我们有：

根据上式，可以计算出第层的,梯度如下：

那么现在问题的关键就是要求出了。

假设第层的已经求出来了，则

可见，关键在于求解

而和的关系其实很容易找出：

可以求出：

讲上式带入上面和关系式得到：

**神经网络训练流程**

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*{(x1,y1),(x2,y2),...,(xm,ym)}*

输出：各隐藏层与输出层的线性关系系数矩阵和偏倚向量

1. 初始化各隐藏层与输出层的权重矩阵和偏移向量的值。

(2-1)

1. 输入设置为
2. *，*进行前向传播算法计算
3. 通过损失函数计算输出层的
4. ,进行反向传播算法计算

(2-2) ,更新第层的：

(2-3)  如果所有,的变化值都小于停止迭代阈值ϵ，则跳出迭代循环到步骤3。

（3）各隐藏层与输出层的权重矩阵和偏移向量