第 6 章 神经网络

6.1 试述将线性函数 $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$ 作为激活函数的缺陷。

（1）由于 $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$ 是线性的。但是神经网络中使用的应该是一个非线性的激活函数，如果在神经网络中使用线性的激活函数，那不管神经网络又多少层，多复杂，最后都可以转化为一个线性回归，这显然不是我们所需要的，也无法实现深度学习功能；

（2）同时线性函数对 0 周围的数值，或者趋近于∞的数值的特性是相似的，这一点我们也不需要，我们需要的激活函数我们更在乎那些处于 0 周围的，在被分为正（大于 0）还是负（小于 0）的界限边的’模糊’的数值。

6.2 以下是几种在神经网络或深度学习网络中常用的激活函数，试总结激活函数所具备的特征，并解释下面几种函数是否适合作为激活。

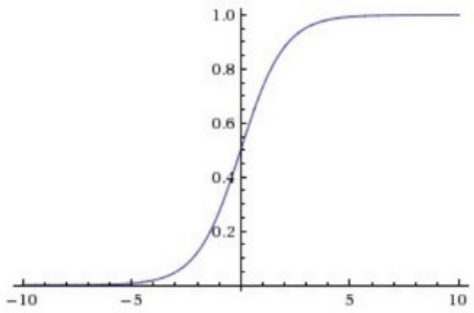
这些神经元单元(可以看成激活函数的性质)应该具有下列基础性质：

- 有界
- 容易求导
- 单调(容易进行凸优化)
- 处理简单(计算方面)

（1）$\mathbf{g(x)} = \frac{1}{1+e^{-ax}}, a > 0$

这个是 sigmoid 函数可以作为激活函数，sigmoid 函数输入一个实值的数，然后将其压缩到 0~1 的范围内

Sigmoid 函数公式为：$\sigma(x) = \frac{1}{1+e^{-x}}$，函数图像如下：



优点为：

- 将所有数据映射成了（0,1）之间的数，很好的表达神经元的激活与未激活的状态，适合二分类。

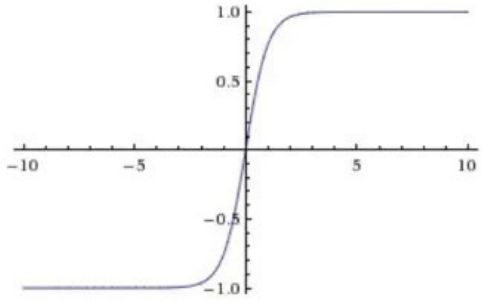
缺点为：

- 存在梯度弥散现象，即 sigmoid 函数很容易饱和，在输入的值很大或者很小的时候，函数梯度接近 0，在反向传播中，会导致导数为 0，权重基本没什么更新
- 计算量大，sigmod 函数要进行指数运算，这个对于计算机来说是比较慢的
- Sigmoid 函数的输出不是 0 均值的，导致后层的神经元的输入是非 0 均值的信号，这会对梯度产生影响，有一种捆绑的效果，使得收敛缓慢。

（2）$g(x)=\frac{1-e^{-ax}}{1+e^{-ax}}=\tanh\left(\frac{ax}{2}\right)$，$a>0$

这个函数是 Tanh 函数，可以作为激活函数，Tanh 函数把输入压缩到（-1,1）的范围，因此它基本是 0 均值的

Tanh 函数的公式为$\tanh(x)=\frac{\sinh(x)}{\cosh(x)}=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$，函数图像为：



优点为：

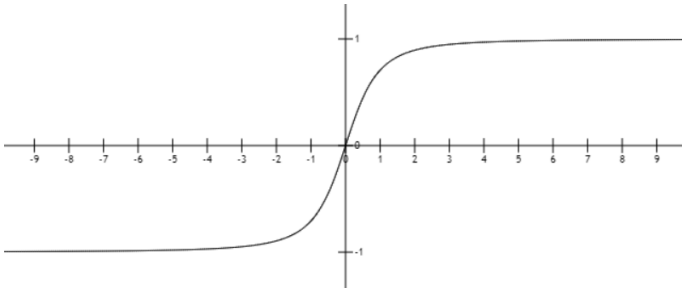
- 它是 0 均值的，解决了上述 Sigmoid 缺点中的第二个，所以实际中 tanh 会比 sigmoid 更常用

缺点为：

- 它还是存在梯度弥散的问题
- 计算量仍然很大，函数要进行指数运算，比较慢

（3）$g(x)=\frac{x}{\sqrt{1+x^{2}}}$

这个函数与 tanh 函数类似，可以作为激活函数，将输入的值压缩到（-1，1）的范围



优点缺点与（2）的 tanh 函数类似，这里不再叙述。

6.3 神经元 j 从其它四个神经元接受输入，它们的值分别为 10，-20，4，-2。神经元 j 的每个突触的权值分别为 0.8，0.2，-1.0，-0.9。计算下列两种情况下神经元 j 的输出。

神经元输出计算过程如下：

- 计算$\sum_{j}^{n} w_{ij}x_{j}=10*0.8-20*0.2+4*(-1)+2*0.9=1.8$
- 设激活函数为 f，则$y_i=f(net_i)$，$net_i=\sum_{j=1}^{n} w_{ij}x_{j}-\theta$

（1） **偏置$\theta=0$，神经元是线性的（即不经过激活函数的处理）。**

$\theta=0$，无激活函数，则$y_i=1.8$

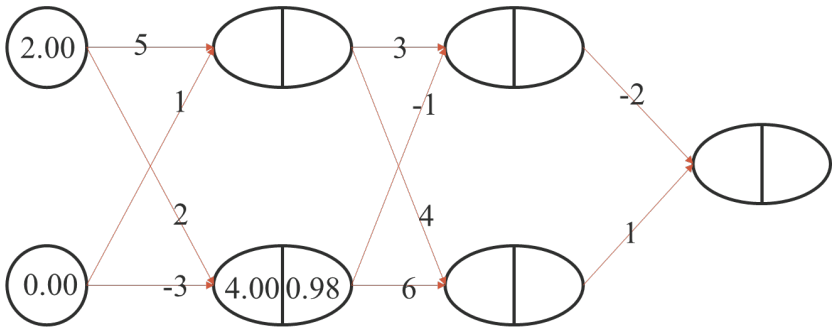
（2） 偏置$\boldsymbol{\theta}=\mathbf{0}$，神经元的激活函数为 sigmoid 函数。

$$y_i=\frac{1}{1+e^{-1.8}}=0.858149$$

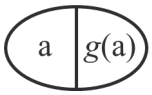
（3） 偏置$\boldsymbol{\theta}=-\mathbf{9}$，神经元的激活函数为 sigmoid 函数。

$$y_i=\frac{1}{1+e^{-(1.8+9)}}=0.99998$$

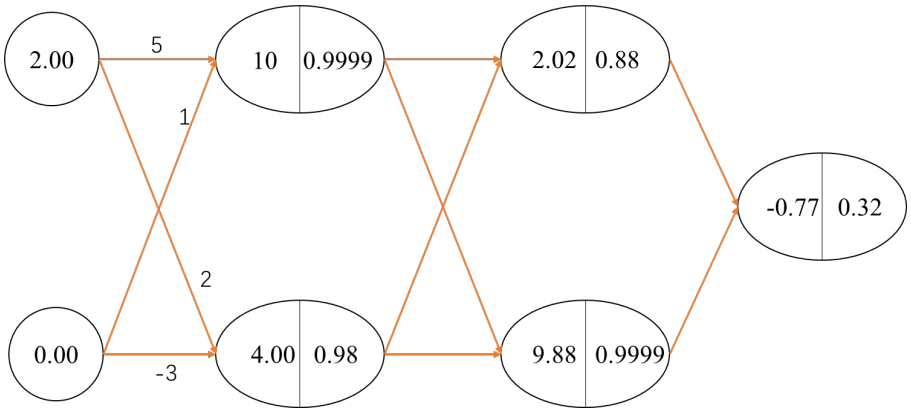
6.4 利用前向传播算法，补全下列网络中结点取值。



注：$g(x)$为 sigmoid 函数

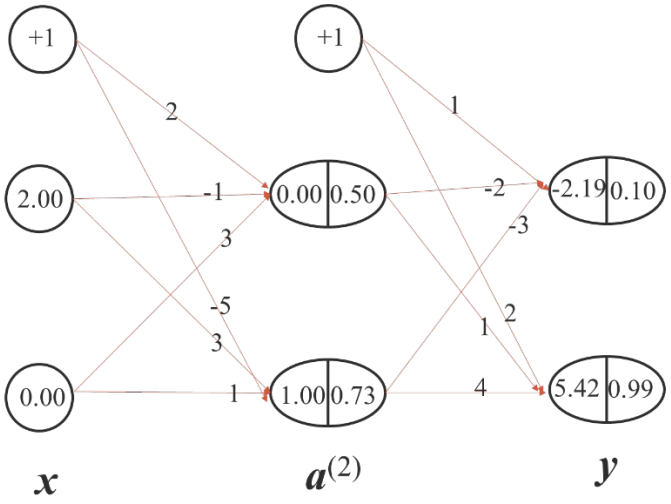


sigmoid 函数为 $\frac{1}{1+e^{-x}}$

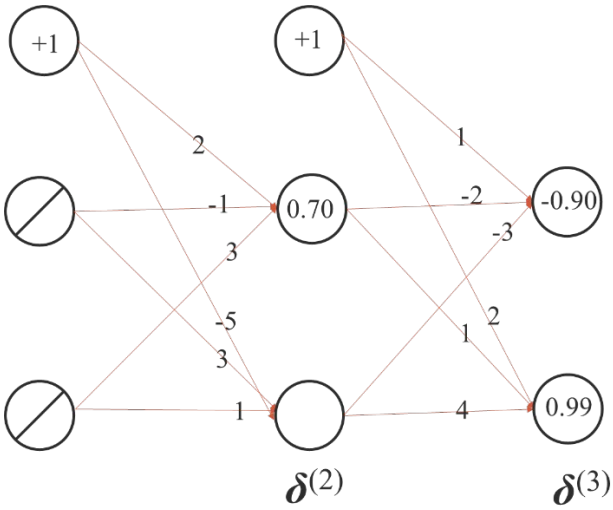


6.5 依据后向传播算法，补全下列网络的结点误差。

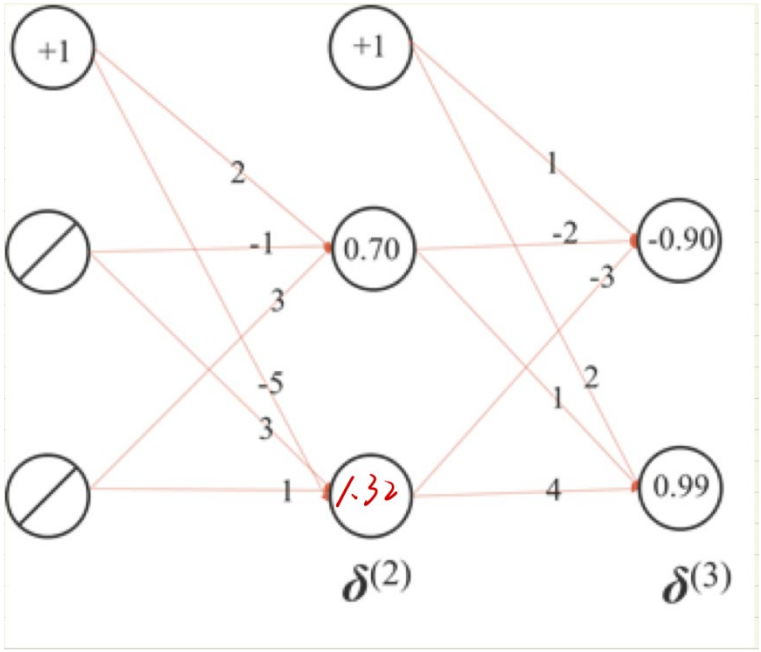
输入特征向量[2.00, 0.00]时，由前向传播算法得到如下结果：



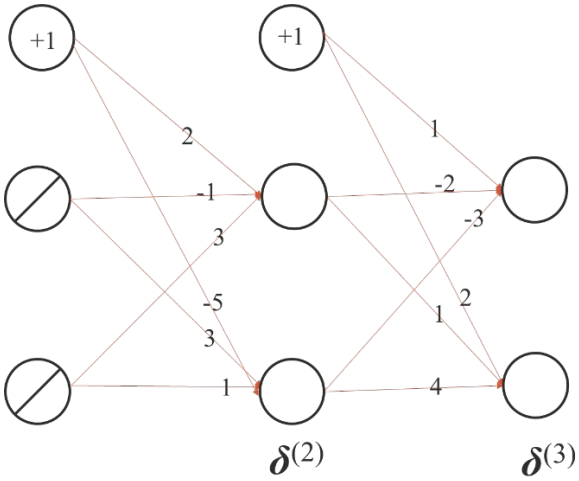
对类别向量 y_1=[1，0]，后向误差传播如下：



计算结果，补全如下：



对类别向量 y_2=[0, 1]，后向误差传播如下：



计算结果，补全如下：