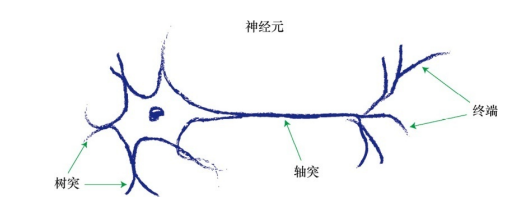
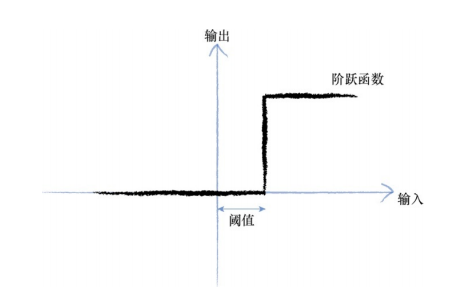
**神经网络原理**

1. **神经网络基本结构：**

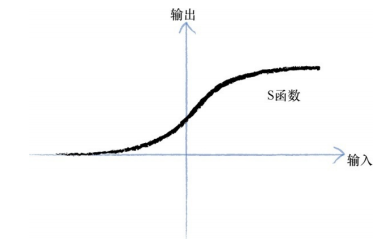
生物大脑的基本单元 -- 神经元，电信号从一端传到另一端：



观察表明，神经元不会立即反应，而是会抑制输入，直到输入增强，强大到可以触发输出。即在产生输出之前，输入必须到达一个阈值。神经元不希望传递微小的噪声信号，只传递有意识的明显的信号。

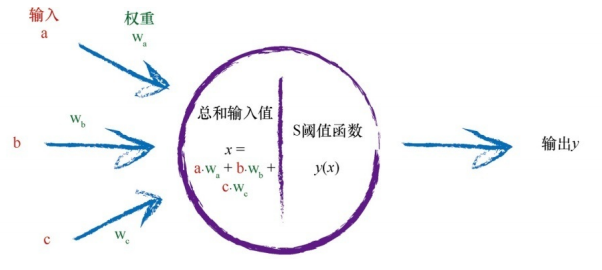


改进阶跃函数为平滑的S函数（更自然，更接近现实）：

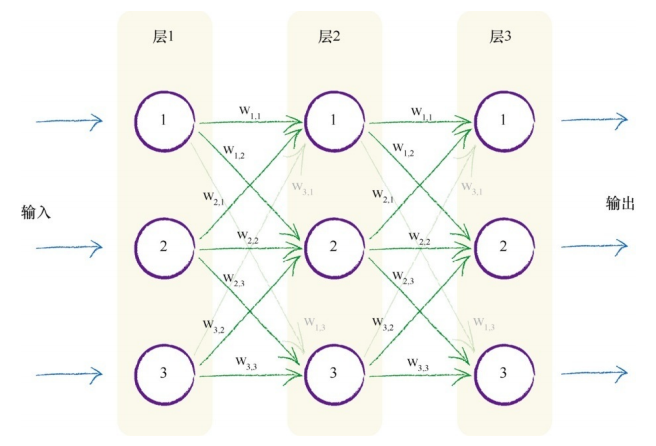


 ,可以写成：sigmoid(x)

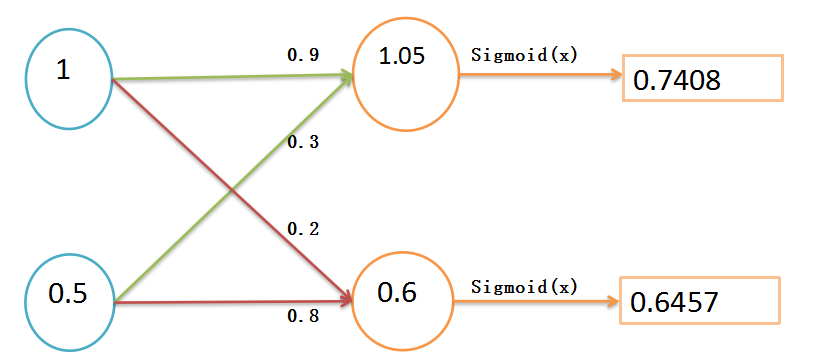
每个神经元接受来自其之前多个神经元的输入， 并且如果神经元被激发了，它也同时提供信号给更多的神经元。



用下图描述这种思想，全部连接上的原因：1.编程方便 2.学习过程中会弱化不相关的连接（连接权重会趋近于0）



走一遍计算过程：



神经网络：通过调整优化网络内部的链接权重改进输出。

1. **BP学习算法优化权重：**

学习过程由信号的正向传播与误差反向传播两个过程组成。

正向传播：输入样本从输入层传入，经过各隐层逐层处理后，传向输出层。若输出层的实际输出与期望输出不符，则转入误差的反向传播阶段。

误差反向传播：将输出误差以某种形式通过隐层向输入层逐层反传，并将误差分摊给各层的所有单元，从而获得各层的误差信号，此误差信号即作为修正单元权值的依据。

误差E就是实际输出和期望输出的差值：

d：准确答案

O：通过该神经网络得到的答案

ONum：输出节点数

根据式将误差E式展开到隐层为：

f()：激活函数

HNum：隐藏层节点数

：隐藏层节点j到输出层节点k的权重

：隐藏层节点j的数值

根据式将误差E进一步展开至输入层为：

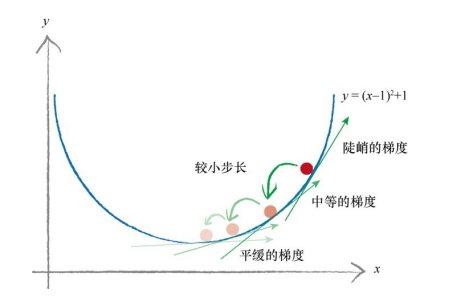
f()：激活函数

INum：输入层节点数

：输入层节点i到隐藏层节点j的权重

：输入层节点i的数值

此时的公式即为代价函数或者损失误差函数。权重W和V都在函数中，可通过调整权重的值来减小误差。



如图所示：往最低点走梯度会变缓，也就是偏导数的绝对值越小。那么咱们就可以用类似的方法来找到误差的低点。也就是梯度下降方法。导数大于0，减少权重，导数小于0，增加权重。

步长（又叫学习率）：每次改变权重的幅度。

求导之前复习一下链式法则：

例：对隐藏层节点j到输出层节点k的权重求导：，其中j=0,1,2,...,HNum ，k=1,2,...ONum。

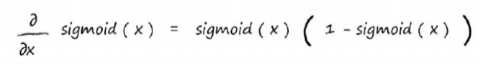
但是对于权重，不需要全部的误差E，因为节点k的输出只取决于链接到k的权重。节点k的误差:

 （1）

 （2）

 （3）

 （4）



 （5）

看着有点费劲，简化下：

，其中j=0,1,2,...,HNum ，k=1,2,...ONum



：学习率。

1. **使用上述方法进行编程：**

S1:初始化数据：定义输入层节点数、隐藏层节点数、输出层节点数、步长。

（权重是随机生成的）

S2：训练：使用BP学习法优化权重，将带有正确答案的数据输入程序中。

S3：测试：只输入数值（不带答案），程序返回计算结果。