Neural Networks-Homework 3

姓名：曹仲　　　　学号：2014310687　　　　班级：自博16

第一题

神经网络结构图像如图 1 所示，

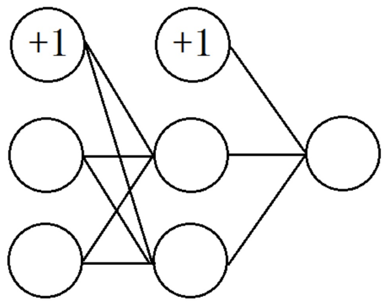


图 1. 神经网络 2-2-1 前向网络结构图

迭代计算：

- 令输入层为第 0 层，其激励函数为线性函数 y = x
- 隐层为第1层，其中第M-1层为输出层，隐层均采用 Sigmoid激励函数，即 $y=1/[1+e^{-x}]$。
- 记第 *l* 层的神经元数目为 N_*l*，其中第0号神经元为阈值单元，其输出恒为1，输出层不含阈值神经元。
- 记第 *l*-1 层中第i个神经元到第*l*层中第j个神经元的连接权为 $\omega_{i,j}^{l-1,l}$
- 从而，第l(l>0)层第j(j>0)个神经元的输入为 $x_j^l = \sum \omega_{i,j}^{l-1,l}y_i^{l-1}$ 。
- 记样本总数为P，NN对应第*p*个样本的第j个输出分量为$t_{j,p}$，则目标函数为

$$E=\sum_{p=1}^{P}E_p=\frac{1}{2}\sum_{p=1}^{P}\sum_{j=1}^{N_{M-1}}(y_{j,p}^{M-1}-t_{j,p})^2$$

其中，神经网络层数 M 为 3，样本总数为 P。

推导过程：前推

第 0 层：

$$\begin{cases} y_{p,j}^0=x_{p,j}^0=u_{p,j}, j=1,\cdots,N_0, p=1,\cdots,P; \\ y_0^0\equiv 1 \end{cases}$$

第 1 层：

$$\begin{cases} y_{p,j}^1=f(x_{p,j}^1)=f(\sum_{i=0}^{N_0}w_{i,j}^{0,1}y_{p,i}^0), j=1,\cdots,N_1, p=1,\cdots,P; \\ y_0^1\equiv 1 \end{cases}$$

输出层：

$$y_{p,j}^{M-1}=f(x_{p,j}^{M-1})=f(\sum_{i=0}^{N_{M-2}} w_{i,j}^{M-2,M-1} y_{p,i}^{M-2}), j=1,\cdots,N_{M-1}, p=1,\cdots,P$$

由目标函数微分，
梯度函数：

$$w_{i,j}^{l-1,l}(k+1)=w_{i,j}^{l-1,l}(k)-\alpha\cdot\frac{\partial E}{\partial w_{i,j}^{l-1,l}}\Big|_{W=W(k)}$$

反向梯度计算：
输出层：

$$\begin{aligned}\partial E/\partial w_{i,j}^{M-2,M-1} &= \sum_{p=1}^{P} \partial E_p/\partial w_{i,j}^{M-2,M-1} \\ &= \sum_{p=1}^{P} \partial E_p/\partial x_{p,j}^{M-1}\ \partial x_{p,j}^{M-1}/\partial w_{i,j}^{M-2,M-1} \\ &= \sum_{p=1}^{P}(-\delta_{p,j}^{M-1}\cdot y_{p,i}^{M-2})\end{aligned}$$

$$\begin{aligned}\delta_{p,j}^{M-1} &= -\partial E_p/\partial x_{p,j}^{M-1} = -\partial E_p/\partial y_{p,j}^{M-1}\ \partial y_{p,j}^{M-1}/\partial x_{p,j}^{M-1} \\ &= (t_{p,j}-y_{p,j}^{M-1})\,\partial y_{p,j}^{M-1}/\partial x_{p,j}^{M-1} \\ &= (t_{p,j}-y_{p,j}^{M-1})\cdot y_{p,j}^{M-1}\cdot(1-y_{p,j}^{M-1}), j=1,\cdots,N_{M-1}\end{aligned}$$

隐层：

$$\begin{aligned}\partial E/\partial w_{i,j}^{M-3,M-2} &= \sum_{p=1}^{P} \partial E_p/\partial w_{i,j}^{M-3,M-2} \\ &= \sum_{p=1}^{P} \partial E_p/\partial x_{p,j}^{M-2}\ \partial x_{p,j}^{M-2}/\partial w_{i,j}^{M-3,M-2} \\ &= \sum_{p=1}^{P}(-\delta_{p,j}^{M-2}\cdot y_{p,i}^{M-3})\end{aligned}$$

隐层依次倒推，其中，

$$\begin{aligned}\delta_{p,j}^{M-2} &= -\partial E_p/\partial x_{p,j}^{M-2} = -\partial E_p/\partial y_{p,j}^{M-2}\ \partial y_{p,j}^{M-2}/\partial x_{p,j}^{M-2} \\ &= -\partial E_p/\partial y_{p,j}^{M-2}\cdot y_{p,j}^{M-2}\cdot(1-y_{p,j}^{M-2}), j=1,\cdots,N_{M-2}\end{aligned}$$

$$\begin{aligned}\partial E_p/\partial y_{p,j}^{M-2} &= \sum_{n=1}^{N_{M-1}} \partial E_p/\partial x_{p,n}^{M-1}\ \partial x_{p,n}^{M-1}/\partial y_{p,j}^{M-2} \\ &= \sum_{n=1}^{N_{M-1}}(-\delta_{p,n}^{M-1}\cdot w_{j,n}^{M-2,M-1})\end{aligned}$$

$$\delta_{p,j}^{M-2}=\sum_{n=1}^{N_{M-1}}\delta_{p,n}^{M-1}\cdot w_{j,n}^{M-2,M-1}\cdot y_{p,j}^{M-2}\cdot(1-y_{p,j}^{M-2})$$

故最终迭代公式为：

$$\begin{aligned}w_{i,j}^{l-1,l}(k+1) &= w_{i,j}^{l-1,l}(k)-\alpha\cdot\partial E/\partial w_{i,j}^{l-1,l}(k) \\ &= w_{i,j}^{l-1,l}(k)+\alpha\cdot\sum_{p=1}^{P}\delta_{p,j}^{l}(k)\cdot y_{p,i}^{l-1}(k)\end{aligned}$$

$$\delta_{p,j}^{l}(k)=\begin{cases}[t_{p,j}-y_{p,j}^{l}(k)]\cdot f'[x_{p,j}^{l}(k)] & l=M-1\\ f'[x_{p,j}^{l}(k)]\sum_{n=1}^{N_{l+1}}\delta_{p,n}^{l+1}(k)\cdot w_{j,n}^{l+1}(k) & l=M-2,\dots,1\end{cases}$$

在代码 BP 0.py 中实现了该算法，并证明了异或问题可以由该题中的神经网络结构进行求解。

第二题

参数BP算法收敛缓慢的主要原因

- BP算法利用梯度信息来调整权值，在误差曲面平坦处， 导数值较小使得权值调整幅度较小，从而误差下降很慢;在曲面曲率大处，导数值较大使得权值调整幅度 较大，会出现跃冲极小点现象，从而引起振荡。
- 神经元的总输入偏离阈值太远时，总输入就进入激励 函数非线性特性的饱和区。此时若实际输出与期望输 出不一致，激励函数较小的导数值将导致算法难以摆 脱“平台”区。
- 由于网络结构的复杂性，不同权值和阈值对同一样本 的收敛速度不同，使得整体学习速度缓慢。

克服BP算法训练缓慢的常用方法

• 改变学习步长。等效于对权值的改变，从而改变误差
曲面的形状，缩短到达极小点的路径而加速收敛。

• 加动量项和改变动量因子。即，$\Delta w_{ij}(n)=\eta\times\Delta w_{ij}(n-1)-\alpha\times\partial E/\partial w_{ij}$。使权值变化更平滑而有利于加速收敛，但 动量因子需适当选择或自适应改变。

• 合适选择神经元激励函数和初始权值、阈值，并对输 入样本的归一化处理。目的是避免“平台”现象的出现。

• 采用合适的训练模式(如逐一式、批处理、跳跃式等)。 避免权值收敛速度的不平衡现象。

• 采用高阶导数信息、最优滤波法和启发式算法。如二 阶导数法、共轭梯度法、准牛顿法、扩展Kalman算法 和delta-bar-delta算法等。

BP算法易陷入局部极小的 原因和改进措施

• 由于不能保证目标函数在权空间中的正定性， 而误差曲面往往复杂且无规则，存在多个分布 无规则的局部极小点，因而基于梯度下降的BP
算法易于陷入局部极小。

• 改进措施主要有:

- 引入全局优化技术;
- 平坦化优化曲面以消除局部极小;
- 设计合适网络使其满足不产生局部极小条件。

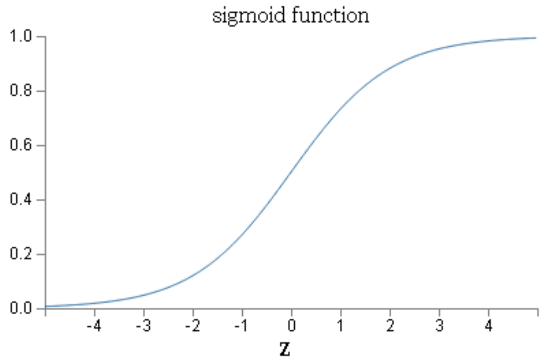
在代码 BP 2.py 中实现了课堂上老师讲的一个例子——仿真实验 2，由于该函数局部最小的很多，容易导致算法收敛和容易陷入局部最小，采用了加动量、改参数、随机化等方法。

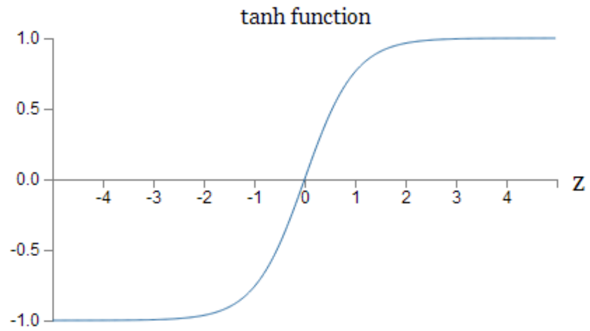
第三题

这两种 sigmoid 函数对比如下，

$$f(x)=\frac{1}{1+e^{-x}} \qquad f(x)=\frac{1-e^{-x}}{1+e^{-x}}$$

两种网络的区别在于将输入的连续实值压缩到的输出范围前者是（0, 1），后者是（-1 1）





事实上，在网络中，前者是对阶跃函数的近似，而后者是对符号函数的近似。在实验 BP 1.py，利用2-2-1 前向网络仿真的时候，两者都可以达到比较好的结果。

BP 算法中，最主要的区别就是导数运算的差异，二者导数分别是

$$f'(x) = f(x)(1 - f(x)) \qquad\qquad f'(x) = 1 - f^2(x)$$

在实验中发现，采用两种 sigmoid 函数对参数的要求差异较为明显，如 BP 1.py，采用前者激励函数，权重取值较大，采用后者激励函数，权重取值较大，对应详细结果参考源代码。

第四题

我所做的方向是模式识别，目前的主要工作是在探索多模态用于深度学习，比如对于既有语音又有图像的信号，能否互相利用特征，使用较少的数据量达到较好的训练效果，前向神经网络潜在的应用包括图像识别、语音识别、翻译等领域，甚至可能逐步揭示数据、信号之间的内在联系。