**课程编号：B080104010**

**人工智能导论**

**作业报告**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **姓名** | **周佳男** | **学号** | | **20154994** |
| **班级** | **软件1507** | **指导教师** | | **郝培峰** |
| **实验名称** | **人工神经网络** | | | |
| **开设学期** | **2017-2018第二学期** | | | |
| **开设时间** | **第1周——第6周** | | | |
| **报告日期** | **2018年4月9日** | | | |
| **评定成绩** |  | | **评定人** |  |
| **评定日期** |  |

**东北大学软件学院**

**一.问题描述**

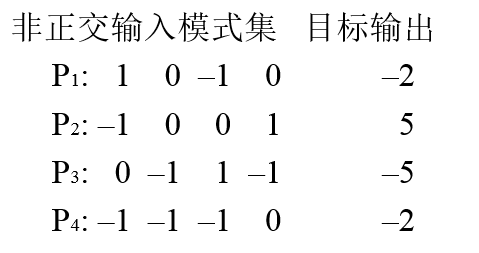
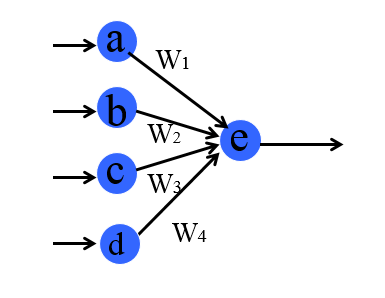
①线性模型学习，给定非正交输入模式集和目标输出；

网络学习规则采用Delt规则：**Δ**wij =**η·**(ti－ai)aj

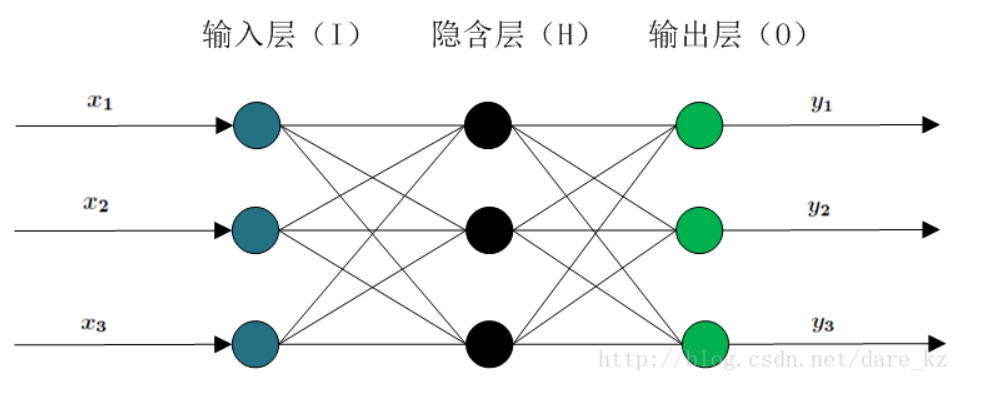
随机置初始权重：W1 = W2 = W3 =0 W4 = 0

置一个较小的学习率： **η** = **0.5**

判断网络学习的准确程度，利用目标值与实际输出误差指标函数：**E = Sqrt(**∑**(tie-aie)2)**



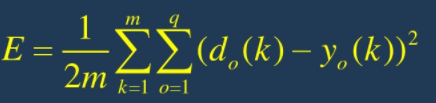
②使用BP神经网络：

  
包括两个过程：正向的计算传播；逆向的误差传播过程。对于本题，有4个输入结点，有1个输出结点，隐含结点设置为2个，单元的激活函数选择S型函数，即Sigmoid函数。形式为：http://latex.codecogs.com/gif.latex?g%5Cleft&space;(&space;x&space;%5Cright&space;)=%5Cfrac%7B1%7D%7B1+e%5E%7B-x%7D%7D，输出为：http://latex.codecogs.com/gif.latex?H_j=g%5Cleft&space;(&space;%5Csum_%7Bi=1%7D%5E%7Bn%7D%5Comega&space;_%7Bij%7Dx_i+a_j&space;%5Cright&space;)，权值的更新Δwij=εδiaj，

当权重连接输出单元时δi=(ti-ai)ai(1-ai)；

当权重连接隐层单元时，δi=ai(1-ai)Σδkwki；

阈值更新：输入层到隐含层：http://latex.codecogs.com/gif.latex?a_k=a_k+%5Ceta&space;H_j%5Cleft&space;(&space;1-H_j&space;%5Cright&space;)%5Csum_%7Bk=1%7D%5E%7Bm%7D%5Comega&space;_%7Bjk%7De_k

隐含层到输出层：http://latex.codecogs.com/gif.latex?b_k=b_k+%5Ceta&space;e_k 全局误差：

**二.设计**

**1.功能设计**

在程序中可以修改样本个数，学习率，样本值等，开始运行后输入学习次数，程序将给出每次的神经网络学习误差，也可以显示每次运行后权重的变化等。

**2.变量设计**

①const int N=4; //权重个数

const int M=4; //样本个数

const double rate=0.5; //η学习率

double p1[M\*N]={1,0,-1,0,-1,0,0,1,0,-1,1,-1,-1,-1,-1,0}; //输入 M=4

double target[M]={-2,5,-5,-2}; //目标输出

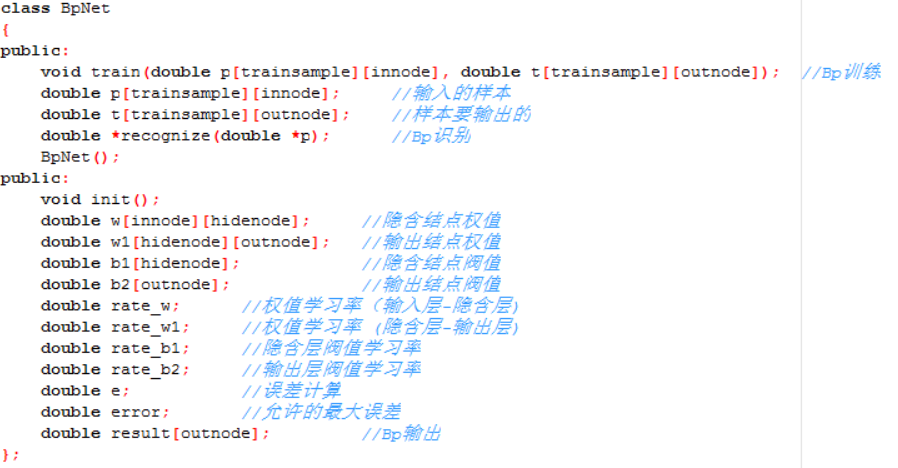
double result[M]={0,0,0,0}; //实际输出

double e=0; //网络误差 e=Sqrt(∑(tie-aie)2)

double w[N]={0,0,0,0}; //权值wi

int count=0; //学习次数

②BP神经网络：



**3.算法设计**

①使用Delt算法进行网络训练，每次学习分别对每个样本进行训练，如对第一个模式对(P1,t1e)进行训练，根据**Δ**wij =**η·**(ti－ai)aj算出每个权重的该变量，更新权重wi=wi+**Δ**wi，在对第二个模式对(P2,t2e)进行训练时，神经网络的实际输出ae已经变化，因此根据新的权重值重新计算ai，然后重复上述步骤改变权重和实际输出值，当所有模式对都训练完毕后，得到了第一次学习后的权重，根据这个新的权重计算每个样本的实际输出，根据误差指标函数E = Sqrt(∑(tie-aie)2)计算本次学习误差，到此第一次学习结束，重复此过程知道满足输入次数。

②BP神经网络，使用单元激活函数为S型函数的学习算法，过程分为正向的计算传播和逆向的误差传播。在某一范围内随机生成权值和阈值，输入一个样本后，在正向的计算传播过程中，计算出隐含层和输出层的输出状态，在逆向的误差传播中，分别更新隐含层与输出层之间的权值和阈值和输入层与隐含层之间的权值和阈值，并计算误差，循环，当误差达到预设精度或学习次数大于设定的最大次数，结束算法。

**三.关键代码(Delt规则)**





**四.实验过程描述**

**1.Delt规则**

本实验输入的是非正交模式集，使用Delt算法进行网络训练，输入学习次数后，得到每次学习误差，可以发现学习误差越来越接近0，因此此方法收敛，如果使用hebb规则学习，可以发现结果不收敛。

①更改样本数量：取样本数量M=3，4，5，6：

M=3时，P1: 1 0 -1 0 🡪-2

P2: -1 0 0 1🡪5

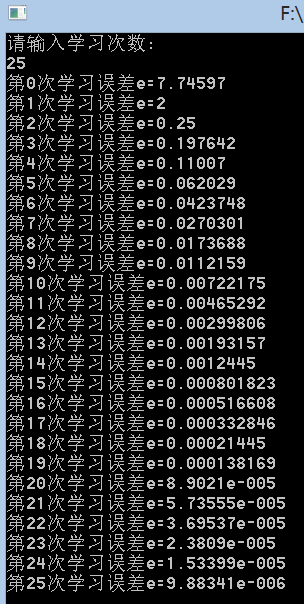
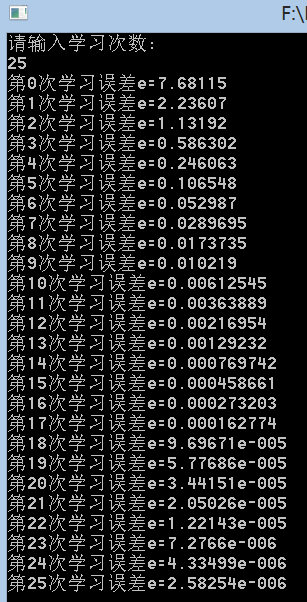
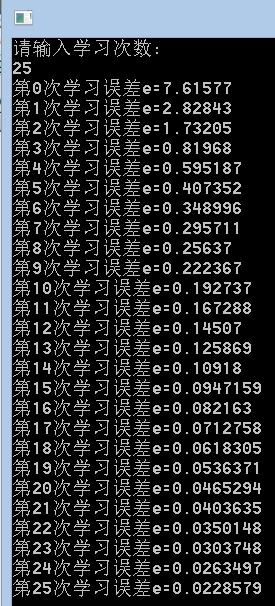
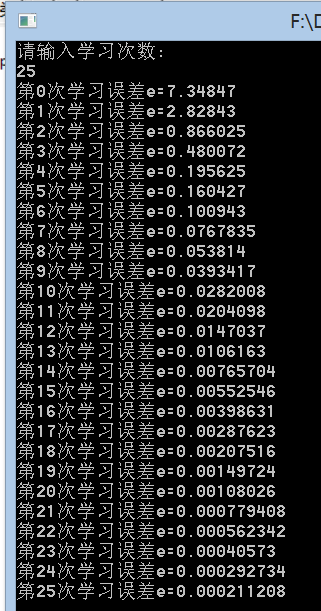
P3: 0 -1 1 -1🡪-5

M=4时，P4: -1 -1 -1 0🡪-2

M=5时，P5: 0 1 -1 0🡪1

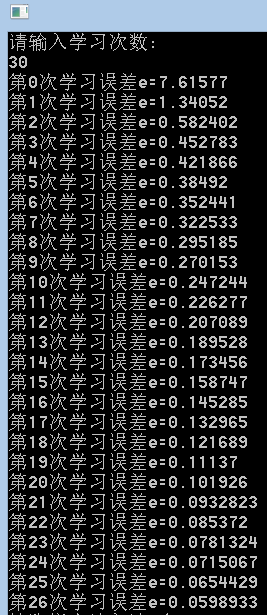
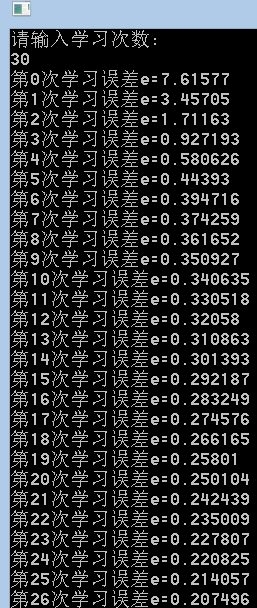
M=6时，P6: 1 1 1 1🡪6 or P6: -1 -1 0 0🡪-1

如下图M由3🡪6：由于初始学习误差不同及图中结果，可见样本数量并不能完全决定人工神经网络的学习速度。



②更改学习率：学习率分别取0.2，0.4，0.5，0.6，0.8：

应选择合适的学习率，如下图学习率为0.8时，学习将不收敛。



**2.BP神经网络**

工作信号正向传递过程（前向传播）：

1、输入层的每个节点，都要与隐藏层每个节点做点对点的计算，计算的方法是加权求和+激活

2、利用隐藏层计算出的每个值，再用相同的方法，和输出层进行计算。

3、隐藏层和输出层都是用Sigmoid作激活函数。

4、起初输入层的数值通过网络计算分别传播到隐藏层，再以相同的方式传播到输出层，最终的输出值和样本值作比较，计算出误差，这个过程叫前向传播(Forward Propagation)。

误差信号反向传递过程

5、利用前向传播最后输出的结果来计算误差的偏导数（前向传播后求偏导），

6、再用这个偏导数和前面的隐藏层进行加权求和

7、如此一层一层的向后传下去（隐藏层间偏导加权求和）

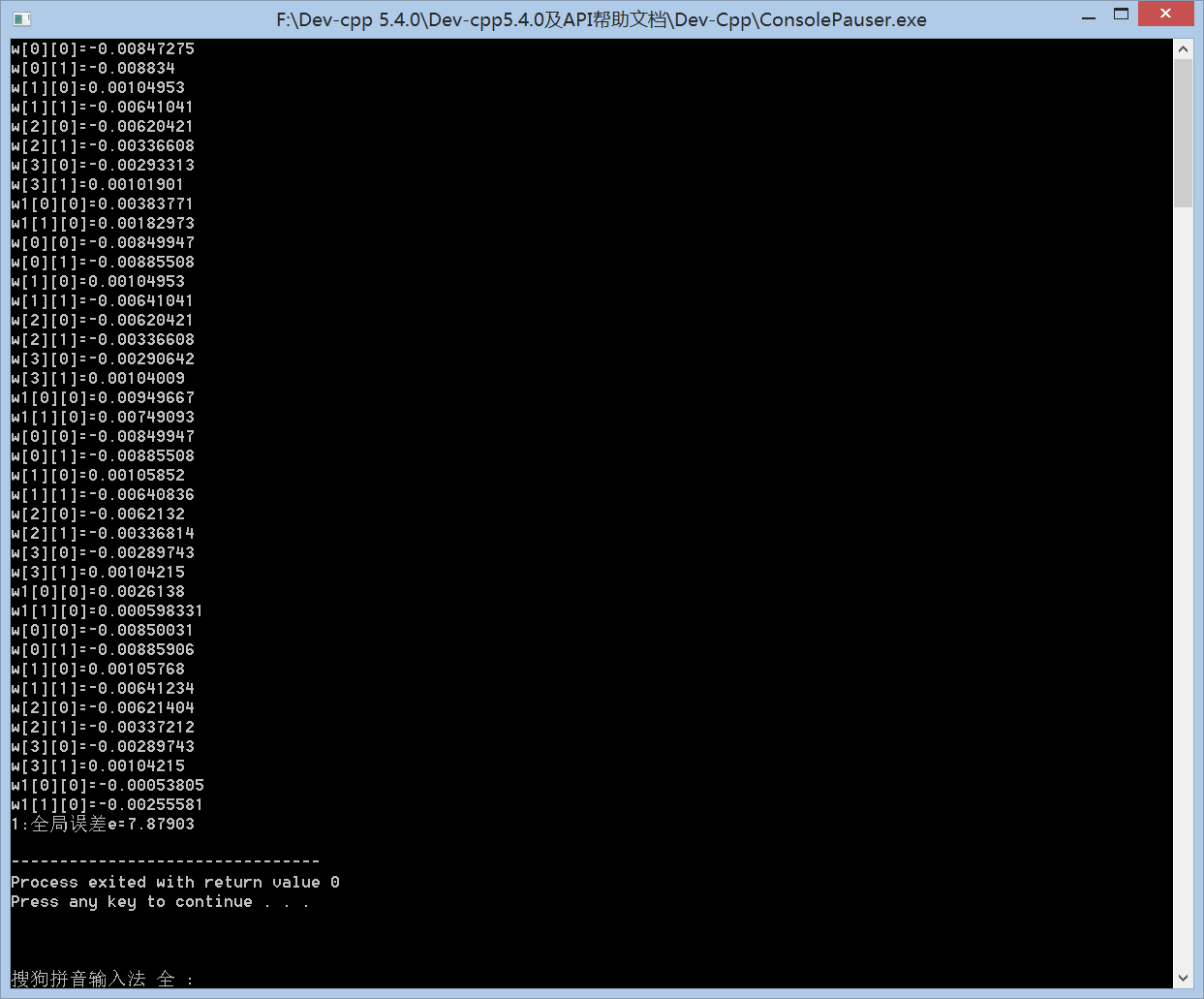
8、直到输入层(不计算输入层)（也就是第一隐藏层到输入层的偏导加权求和）

9、最后利用每个节点求出的偏导数来更新权重。

程序初始化学习率为0.01，权值和阈值初始随机范围[-0.01，0.01)

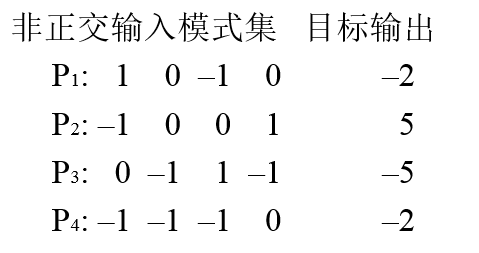
我们进行一次训练(输入4个样本)后，每次的权重变化，全局误差结果如下：

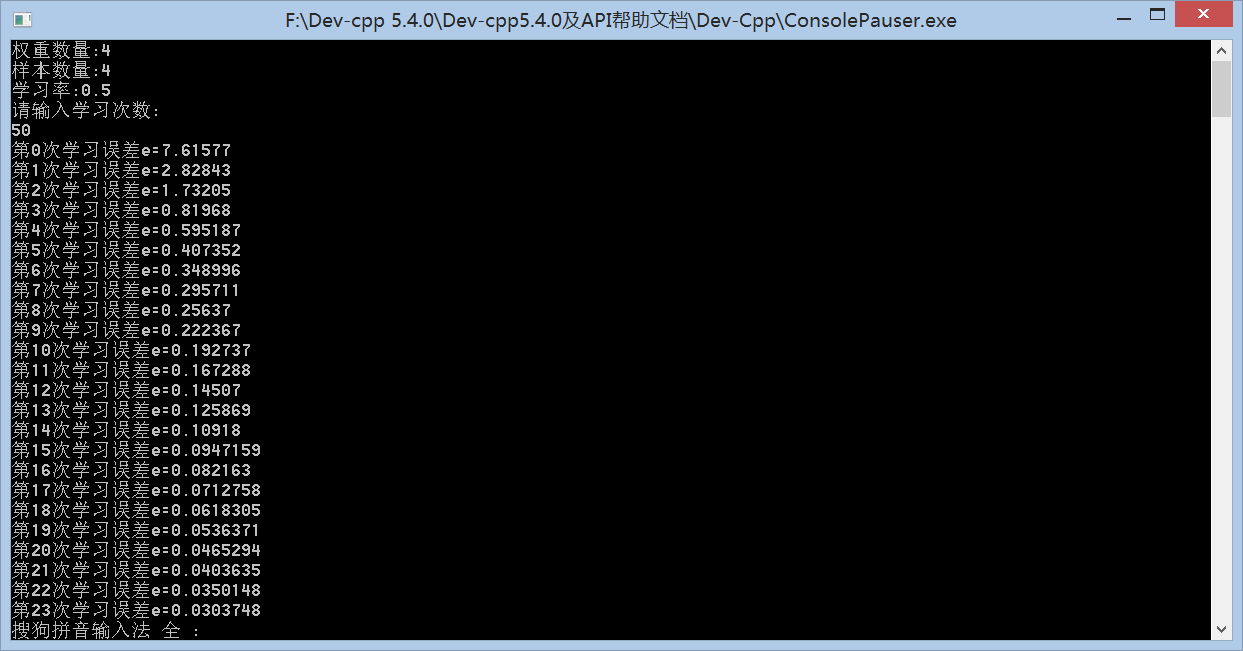
其中w[i][j]代表隐含结点的权值，w1[i][j]代表输出结点的权值，本网络仅含有一层隐含层，及为三层BP网络：



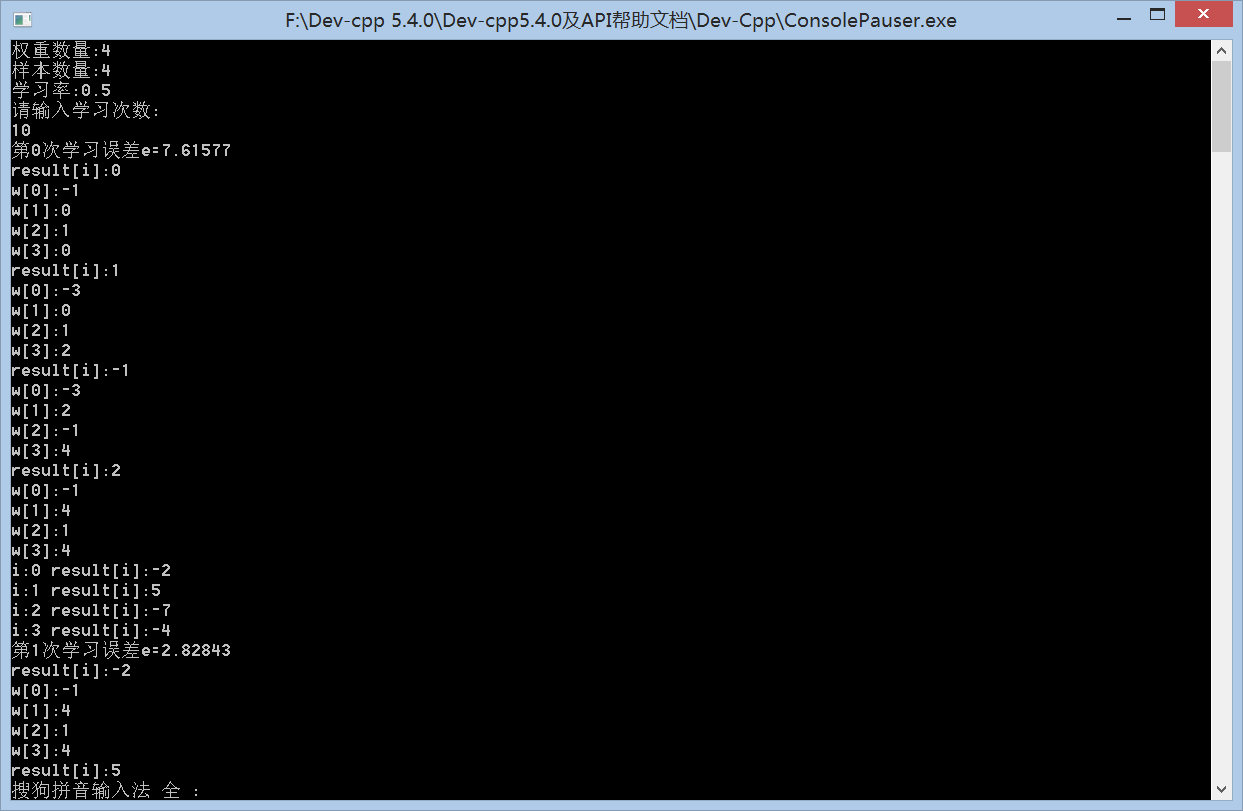
**五.主要功能界面截图**

①N=4，M=4

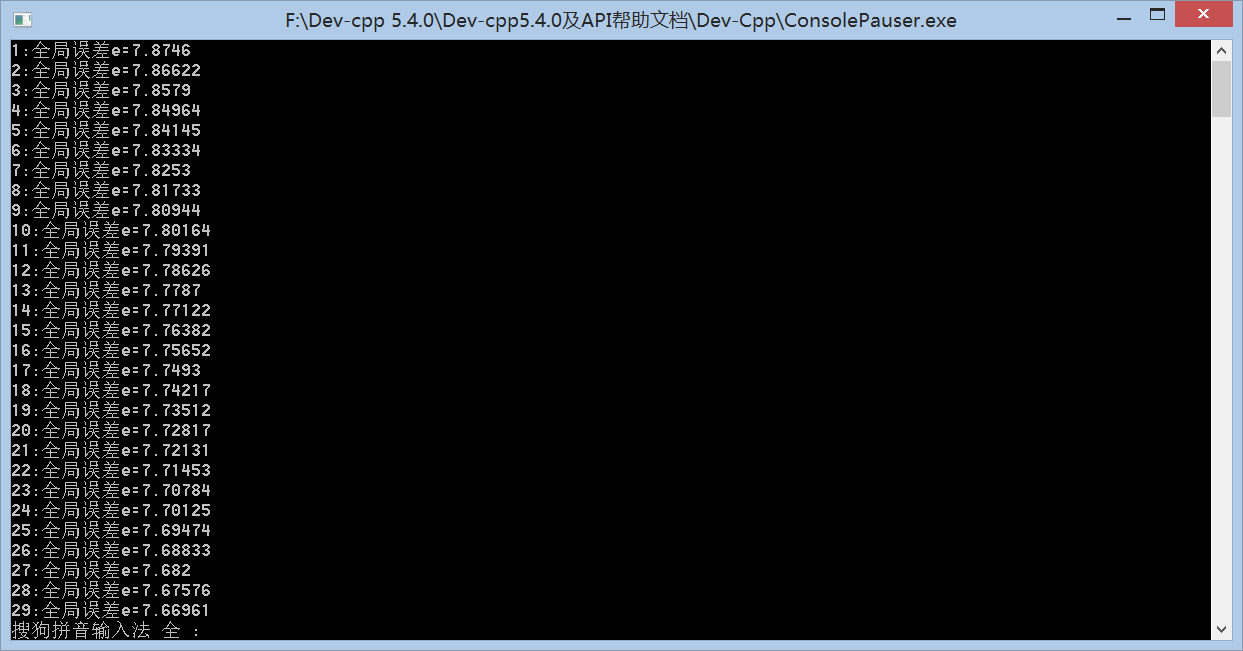


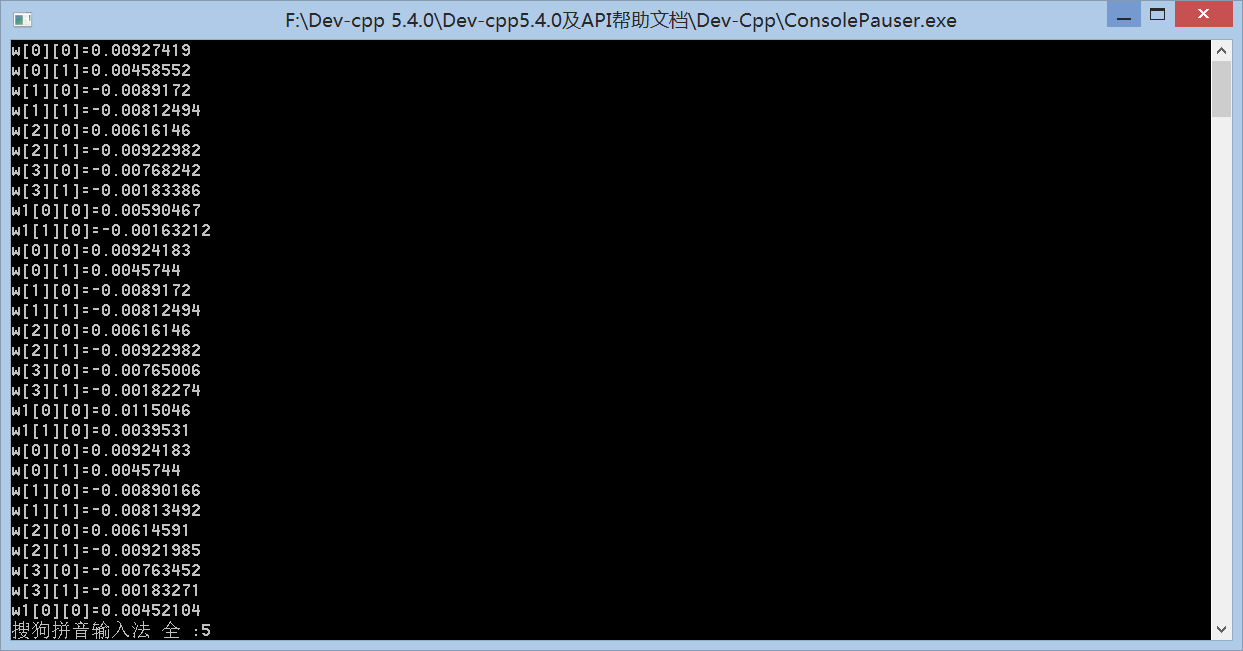


学习的具体过程(去掉函数delt中的代码注释)，W[i]代表权重值，result[i]代表实际输出：



②BP神经网络：(学习率为0.01，权值和阈值初始随机范围[-0.01，0.01))





**六.参考资料**

1. CSDN博客 BP神经网络计算过程详解

https://blog.csdn.net/dare\_kz/article/details/77603522

CSDN博客

1. CSDN博客 简单易学的机器学习算法——神经网络之BP神经网络

https://blog.csdn.net/google19890102/article/details/32723459

1. 百度文库 BP神经网络算法原理

https://wenku.baidu.com/view/b621d2cc0722192e4436f627.html