

山东大学机器学习课程

实验报告

——实验五：手写bp神经网络实现数字识别

姓名：陈竞帆

学院：软件学院

班级：软件八班

学号：201500150180

**一、实验目的：**

（1）熟悉python实验软件及相关函数

（2）学习全连接神经网络并学习反向传播算法

（3）利用自己设计的神经网络对mnist数据集进行训练并测试

**二、实验环境：**

（1）硬件环境：

英特尔® 酷睿™ i5-7500U 处理器

256 GB PCIe® NVMe™ M.2 SSD

8 GB LPDDR3-1866 SDRAM

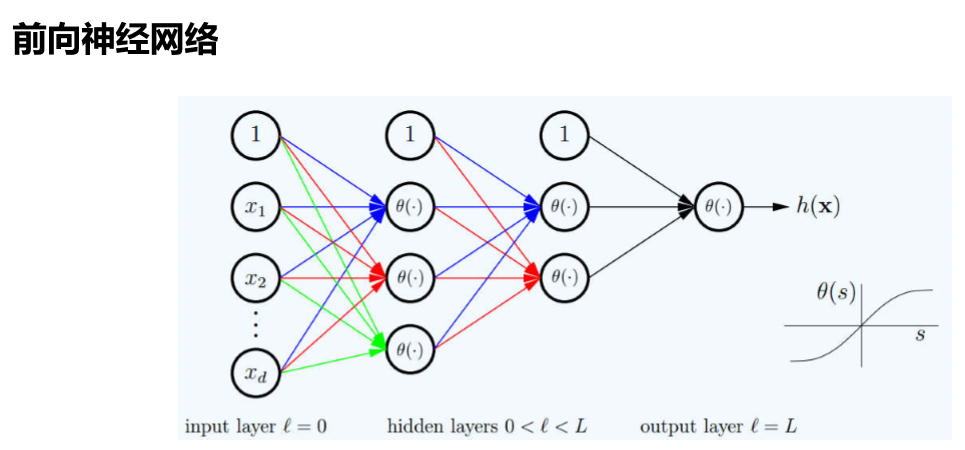
（2）软件环境：

Windows10家庭版64位操作系统

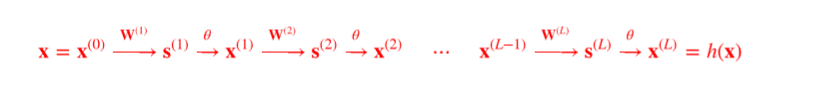
Python3.6

**三、实验内容**

**3.1 全连接神经网络以及反向传播算法概述**

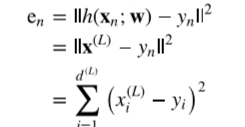


**神经网络需要进行一次前向传播，然后设计一个损失函数，对损失函数求解最优解，利用梯度下降方法，得到最优解下的权值，这是反向传播算法的整体概述。**



**前向传播算法流程如上图所示，从输入端开始，逐步利用权值初始值，进行下一层的计算，最后得到一个output。**

**接下来需要定义一个损失函数，对于某一个样本：**



**在我们的实现中，对每一个样本求一次梯度，当所有的样本都求了一次梯度后，则记为一次迭代，迭代的次数可以自己设定。**

**下面是对bp算法的介绍：**

**我们利用第一次的前向传播得到了一个输出值，我们需要对我们的损失函数进行一个最小值的求解。**

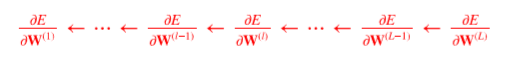


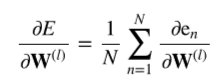
**其中yita是学习率，自己设置，在实验中我们设置为0.05，在这种学习率下效果较好。**

**接下来是对梯度的计算，即损失函数对权值求偏导。**

**我们对权值的求偏导，即对每一层之间的权值分别求偏导。**

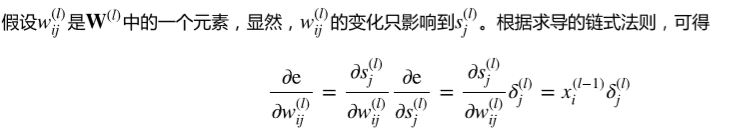
**那么整体的流程就是和前向传播是相反的：**



**其中：**

**因此我们假设我们只对每一个样本分别进行反向传播算法的求解，那么只需要求解：**

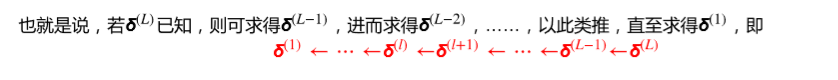




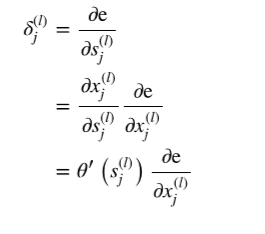
**因此我们需要求解一个敏感度向量：**

**其中S为每一层的输入，x为每一层的输出**

**由于x的值我们根据前向传播已经求解得到，那么我们求解的重心将会放在对敏感度向量的求解。**

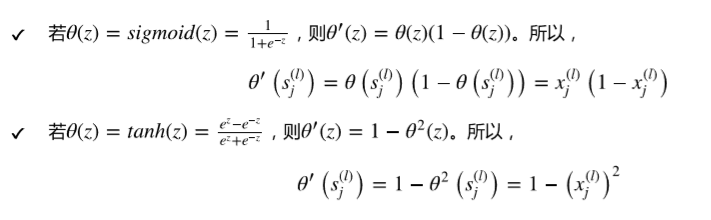


**敏感度向量的求解也是从后往前的。**



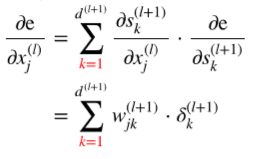
**我们再次利用链式法则，可以得到如上的公式。**

**Theta为我们采用的激活函数：**

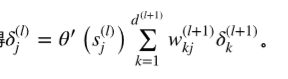


**那么我们的重心将会放在如何求解对x的偏导。**

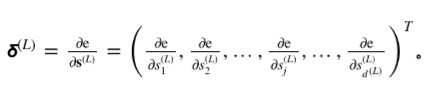


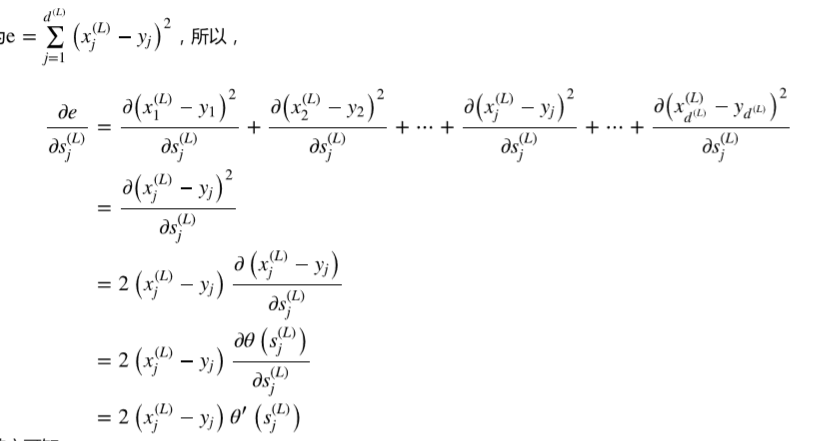


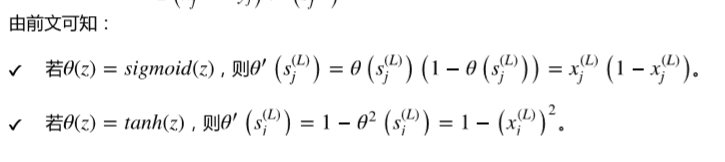
**所以，整个过程的重点是对敏感度向量deta的求解。**



**从公式中可以看出，我们首先需要求解最后一层的deta层的值。**







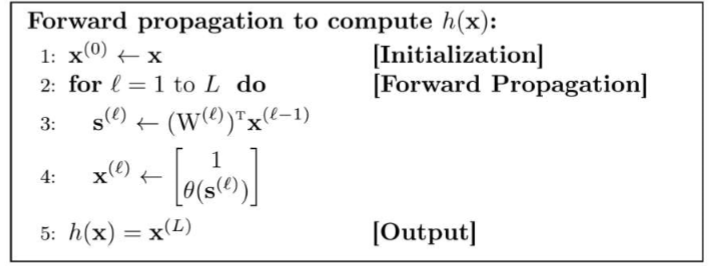
**因此最后一层的deta可以计算得出，从而得出所有的deta值。**

**有了deta值后，我们的每一层的梯度向量=每一层deta值\*前一层的输出值x**

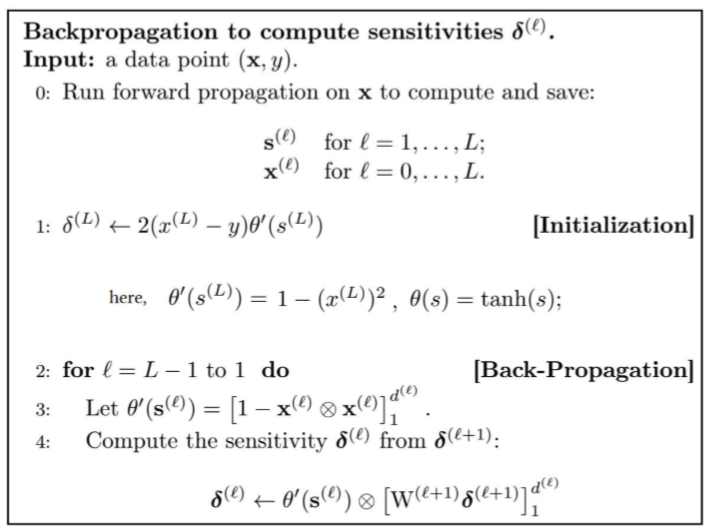
**那么每一层的权值即可根据梯度向量和学习率进行变化。**

**3.2前向传播和反向传播算法**

设计程序的伪代码，实现图1所示：



前向传播算法



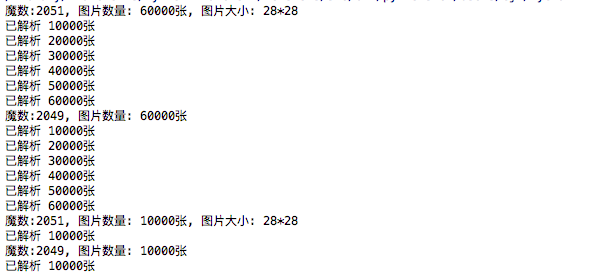
反向传播算法

**图1.两种梯度下降法的伪代码**

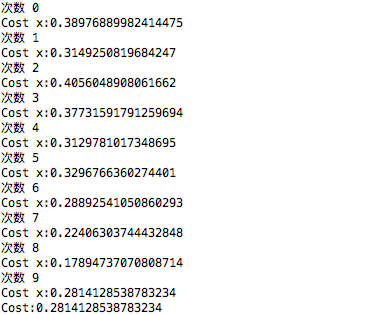
**四、实验结果**

**在我们的实验中，我们需要对数据集的label进行相应的设置，因为label是0-9的数字，我们需要将他转换成1\*10的数组，对相应的位置的数字标为1，其余的数字标为0**

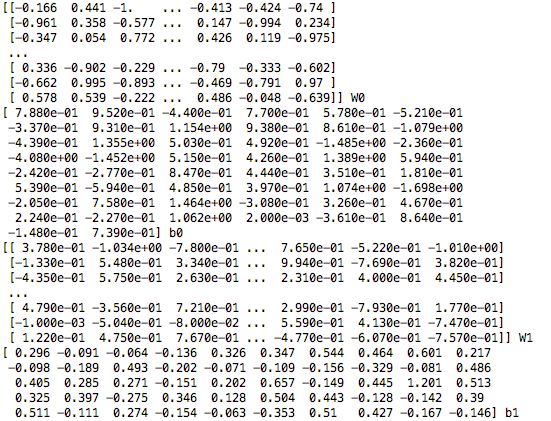
**我们最后设置了两个网络结构，一个是784\*50\*50\*10，另一个是784\*50\*10**



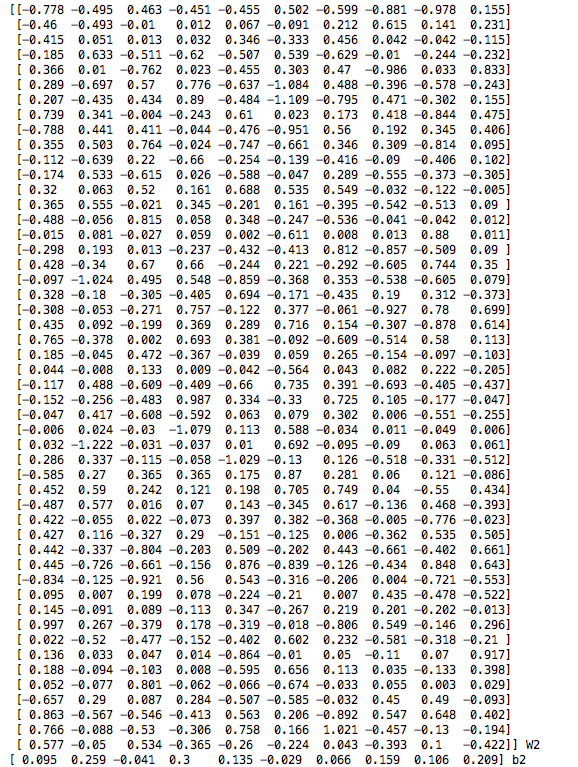
**图2. 对mnist数据集的读取结果展示**



**图3. 利用两层hidden层结构的网络10次迭代每一次的损失结果**



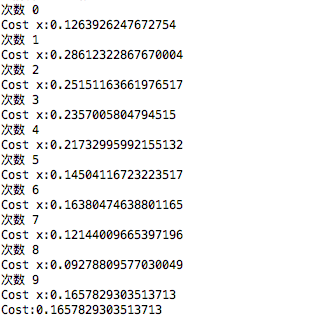
**图4. 两层hidden层结构的网络的部分参数**



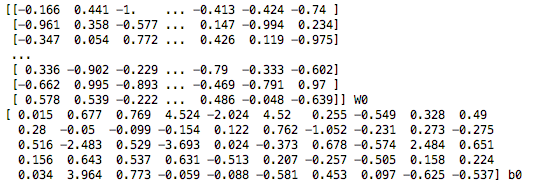
**图5. 两层hidden层的神经网络的部分参数**

C:\Users\95171\Documents\Tencent Files\951718744\Image\C2C\3c82fd46-1988-48a2-891d-7f85d89783db.JPG

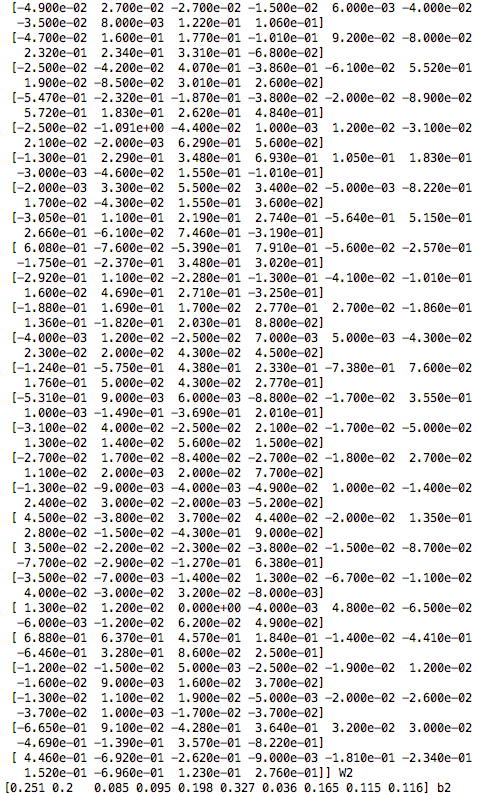
**图6. 两层hidden层对测试集测试的准确率结果**



**图7. 利用一层hidden层结构的网络10次迭代每一次的损失结果**



**图8. 一层hidden层结构的网络的部分参数**



**图9. 一层hidden层的神经网络的部分参数**

C:\Users\95171\Documents\Tencent Files\951718744\Image\C2C\cfe34d09-e4a8-43f7-b323-c24a50a42b64.JPG

**图10. 一层hidden层对测试集测试的准确率结果**

**五、总结与归纳**

**这次的实现较难，需要手写一个bp神经网络。我一开始使用的是python中的面向对象思想，将unit、layer、neural network分别建立class，然后利用list进行传播。但是在实验的训练过程中发现list的效率非常低下。因此采用了以每一层中的权值矩阵为主，直接对矩阵进行运算。在运算的过程中，是需要手工设置层数和层中节点的数量。我们采用了一条样本进行一次权值更新。由于实验过程中发现错误率并没有因为迭代次数的增加有明显的改善，因此迭代次数设置的较小，为10次。我们设置了两种网络结构，一种是50\*50的隐层，一种是50的隐层，最后的准确度差不多，都为90%左右。反而是学习率对最后结果的影响较大。**

**由于是自己写的网络，对神经网络以及bp算法有了深刻的认识，而不仅仅是公式上的了解。巩固了基础。**