智能计算体系结构课程lab1实验报告

作者：北京航空航天大学计算机学院 陈胤佳

学号：19373383

一、实验目的与要求：

本次实验利用Python进行了简单的深度神经网络（DNN）的搭建，并使用给定的训练集和测试集，Mnist官方数据集进行训练与测试。内部细节采用了业界最为经典的几种算法与函数，旨在对于神经网络结构有详尽的了解与掌握，以便于服务于课程后期智能计算体系结构的搭建。

二、实验环境：

IDE：PyCharm 2021.2.2 (Professional Edition)；

Environment：Python 3.9；

OS：Windows 10 Professional

三、实验内容与步骤

1. 分析思路：

本次实验的重点在于理解并掌握神经网络的结构与其内部算法的原理。深度神经网络由若干感知机组成；单个感知机输出结果为：

其中为感知机的输入，为输出，为每个输入对应的权重，为神经激活函数。

深度神经网络的主要结构包括输入层1层、输出层1层、与若干隐藏层，每层均包含了若干个感知机模型，层与层间的每个感知机模型都相互连接，每条相连的边都拥有各自的权重，每个感知机模型节点都拥有各自的偏移量。

前向传播算法：结成完整的网状模型后，当位于输入层给定一个输入向量（即向量每个分量对应输入层每个感知机的输入值），经过网络的层层计算传播，可以在输出层获得一个确定的输出向量。

反向传播算法：由于深度神经网络学习的本质是根据大量的输入输出已给定的数据集对于网络内部如权值与偏移量等数据进行不断的修正与训练，从而使得其获得新数据时能够根据已训练的模型生成与预期更为相符的结果，因而，将输出结果与预期结果进行比较，并反向不断各权重与偏移量的过程则显得尤为重要。经典的反向传播算法中采用均差法来度量损失：

其中，为网络第层的输出，为整个网络所预期的真实输出。

据此，进一步计算权值矩阵中每一个权值对该损失的贡献，并通过求梯度值获取每个权值所应当改变的大小，并对权值进行更改。重复此过程，直至损失函数的结果降低至一定范围内，即误差被认定为正确，则该数据集训练完毕。

2. 设计方案

在课程组提供模板项目（BP-Mnist-Numpy-Template）的基础上进行修改以及补全。

模板结构：

main.py：主函数入口，包含了读取数据集文件的过程，以及train()方法与evaluate()方法，分别用于训练与测试。

utils.py：包含了load\_mnist()方法，用于读取本地的数据集文件，并返回所读取到的向量。

func.py：为笔者补充的文件，其中包含了sigmoid()、sigmoid\_diff()两个方法，分别代表神经激活函数与其导数。

model.py：包含了neuralNetwork类，其中所包含的构造器方法、forward()方法与backpropogation()方法分别是要填充核心代码的部分。

1. 构造器方法：应包含神经网络的自身属性：各层状态机的偏移向量、、各层输入的权值矩阵、以及学习率；另外由于前向后向传播方法的分离，还需要存储各层线性输出向量、真输出向量、以及各层的梯度向量；另外，为保证该网络结构的代码可拓展性良好，保存了一些基本属性信息如总层数、各层感知机节点数量等常值（本实验中输入输出由具体场景确定，输入为图片像素值，重塑为784个域所组成的一维向量，因此输入层由784个节点构成；输出为分别对应0-9共10个数字的独热编码，因此输出层由10个节点构成；隐藏层选定为1层、400个节点）。

在初始化一个DNN对象时，应对各权值矩阵以及各偏移量向量进行恰当的随机初始化，供后期进行训练。

1. forward()方法：根据输入向量进行逐层计算，并保存各层的线性输出向量以及真输出向量，即：

其中，是第层的真输出向量，是第层的线性输出向量，为整个神经网络的输入，为网络层数，为第层的权值矩阵，是第层的偏移量向量，是非线性神经激活函数。

1. backpropogation()方法：根据输出值与预期值的差经损失函数计算而得的损失，计算最后一层的梯度向量，并进而进行逐层计算，保存各层的结果。根据数学推导，最终能够得到的递推公式为：

另外，需要对每层的权值矩阵与偏移量向量进行更新：

其中，是第层的梯度向量，是在该输入下的期望输出。

根据设计方案在模板中实现Python代码。

3. 技术实现细节与遇见问题

1) 问题一：运行过程中矩阵相乘格式不匹配问题。

在运行后有若干关于矩阵格式不相符导致运算失败的问题，主要问题有两点：第一，没有严格按照数学推导中的矩阵顺序书写代码，导致矩阵相乘顺序错误。第二，方法输入的格式为一维矩阵格式如[10,]，不能直接与二维矩阵相乘，需要先将其转化为二维格式[10,1]，二者含义不同。对两点进行修正后运行成功。

2) 问题二：运行结果持续输出相同值（输出如图1，损失量不收敛，此处仅展示三行输出，下同）。

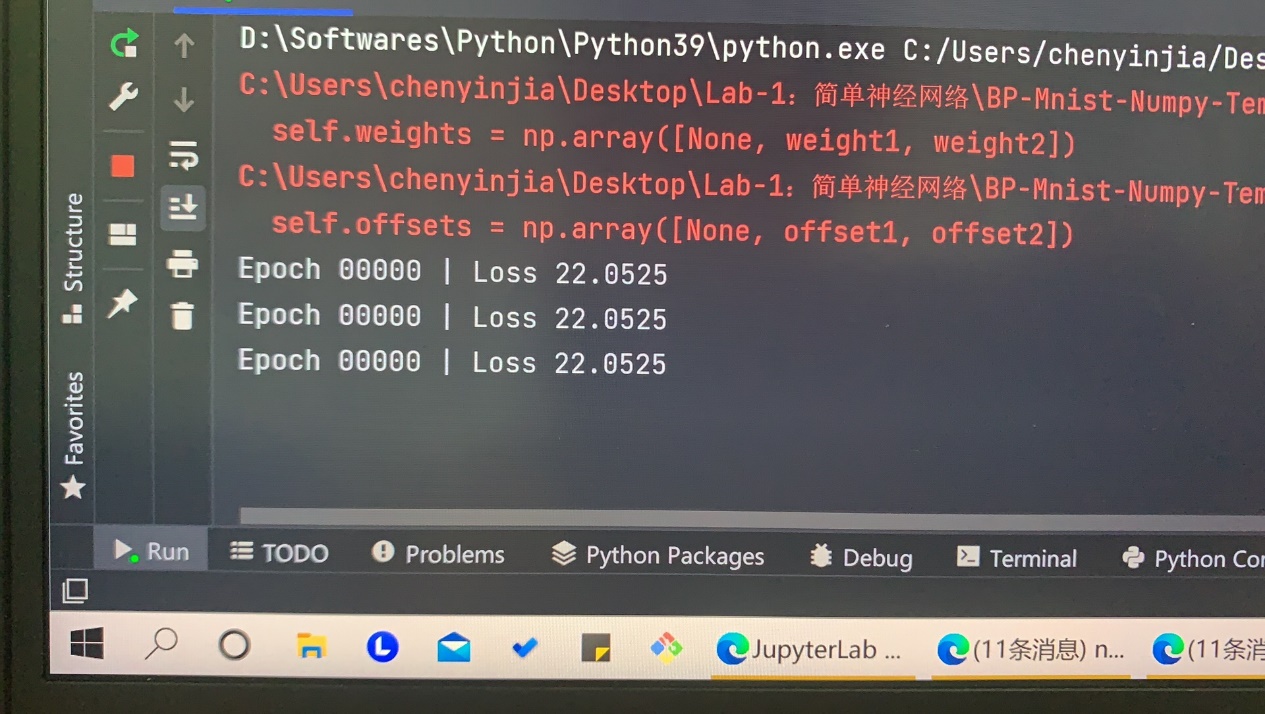


图1

主要原因在于，权值矩阵与偏移量矩阵初始量设置虽然值限定在了0、1之间，但太过随机，导致在计算过程中会由于精度问题丢失位数，且收敛过慢，效率过低。因此需要在初始化过程中加以限制，由np.random.randn(hidden\_nodes, input\_nodes)改为：np.random.randn(hidden\_nodes, input\_nodes) \* np.sqrt(1 / hidden\_nodes)，使数据更为紧凑，贴近真实值并加快收敛速度。

四、实验结果

输出结果如图2：

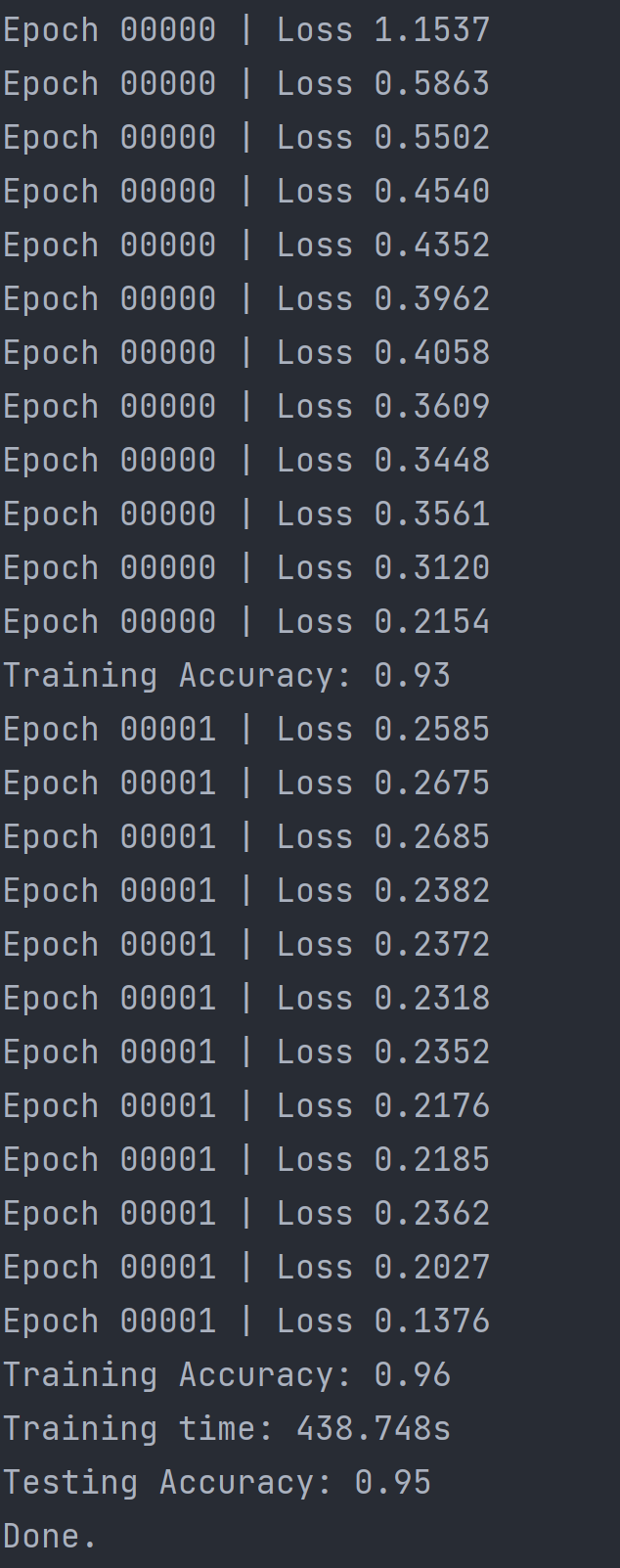


图2

可以观察到第一遍6万次训练中，损失由第一个5000次后的1.1537下降到了0.2154，第一次测试结果正确率为93%；第二遍6万次训练中，损失由第一个5000次后的0.2585下降到了0.1376，第二次测试结果正确率为96%。准确率较高，训练效果较为良好。

五、分析与讨论

本次实验利用py代码，根据已有算法实现了一个简单的神经网络，并利用所提供的数据集对该网络进行了训练与测试。整体而言更为重要的是对于网络结构的理解与掌握，真正在实现过程中遇到的困难实际上并不多，实际结果整体而言也是喜人的。希望本次实验能够为日后课程的lab打下坚实的基础。