**深度学习与神经网络**

**全链接神经网络**

**实验报告**



姓名：张卓倚帆

学号：2021213325

班级：2021219109

学院：人工智能学院

日期：2023年10月20日

目录

[**一、 概述** 3](#_Toc148726831)

[**1. 实验内容** 3](#_Toc148726832)

[**2. 实验环境** 3](#_Toc148726833)

[**二、实验原理** 3](#_Toc148726834)

[**1. MNIST数据集理解及处理** 3](#_Toc148726835)

[**1.1 MNIST数据集** 3](#_Toc148726836)

[**1.2 数据处理** 4](#_Toc148726837)

[**1.3 验证集划分** 4](#_Toc148726838)

[**1.4 数据预处理** 4](#_Toc148726839)

[**2. 搭建全链接神经网络** 5](#_Toc148726840)

[**2.1 激活函数** 5](#_Toc148726841)

[**2.2 前向传播** 5](#_Toc148726842)

[**2.3 反向传播** 6](#_Toc148726843)

[**2.4 损失函数** 7](#_Toc148726844)

[**2.5 模型训练及预测** 8](#_Toc148726845)

[**三、结果与分析** 8](#_Toc148726846)

[**结果：** 8](#_Toc148726847)

[**分析：** 10](#_Toc148726848)

[**四、拓展与创新** 10](#_Toc148726849)

[**1. 优化算法** 10](#_Toc148726850)

[**1.1mini-batch GD** 10](#_Toc148726851)

[**1.2 SGD** 11](#_Toc148726852)

[**2. 损失函数** 12](#_Toc148726853)

[**3. 正则化** 12](#_Toc148726854)

[**五、遇到的问题** 13](#_Toc148726855)

[**1. 数据处理** 13](#_Toc148726856)

[**2. 参数优化** 13](#_Toc148726857)

[**六、总结** 14](#_Toc148726858)

[**七、附录** 14](#_Toc148726859)

**一、 概述**

**1. 实验内容**

（1）设计一个基于全链接神经网络的图像分类器，完成手写数字识别分类任务。

（2）理解基本的图像识别流程及数据驱动的方法（训练、预测等阶段）

（3）理解训练集/验证集/测试集的数据划分，以及如何使用验证数据调整模型的超参数。

（4）实现一个**Softmax**分类器

（5）实现一个**全连接神经网络**分类器

（6）理解不同的分类器之间的区别，以及使用不同的更新方法优化神经网络

**2. 实验环境**

语言：python3

引用的库：numpy用于搭建神经网络，matplotlib用于绘图

IDE：Visual Studio Code

硬件：Windows10系统，CPU执行

数据集：MNIST数据集

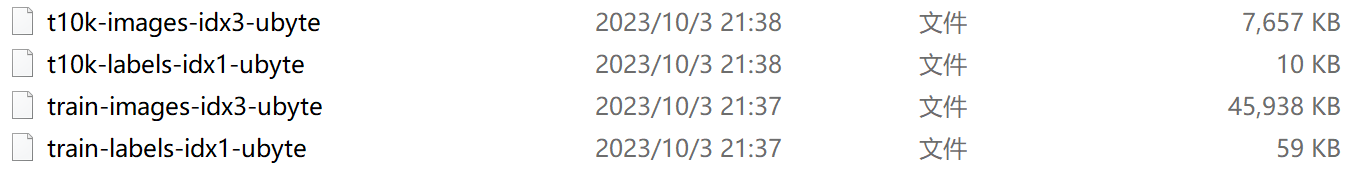
**二、实验原理**

**1. MNIST数据集理解及处理**

**1.1 MNIST数据集**

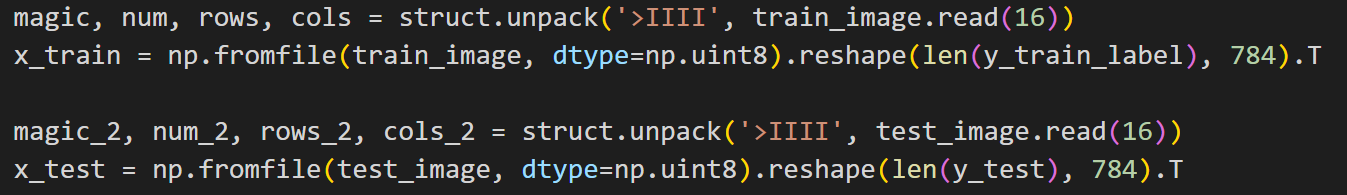
MNIST数据集分为训练图像和测试图像。训练图像60000张（其中5000张为验证集），测试图像10000张，每一个图片代表0-9中的一个数字，且图片大小均为28\*28的矩阵。

从官网下载数据集：



**1.2 数据处理**

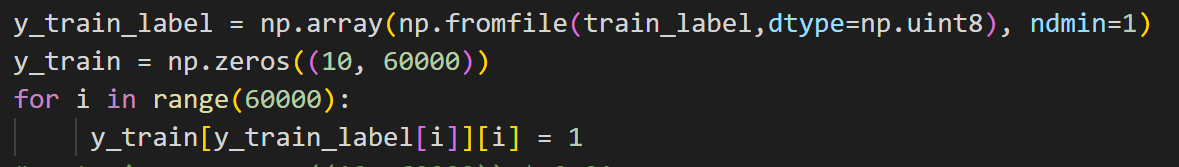
（1）运用struct模块和numpy库将二进制的MNIST中的数据转化为numpy数组。



训练集数据(784x60000) 训练集标签(1x60000) 测试集(784x10000) 测试集(1x10000)

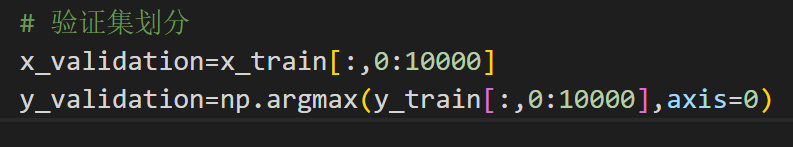
（2）one-hot编码

对训练集标签进行one-hot编码



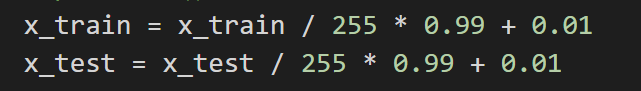
**1.3 验证集划分**

从训练集中划10000个数据做验证集：



**1.4 数据预处理**

将图像像素值进行归一化处理：



归一化将图像的像素值缩放到一个较小的范围，通常是[0, 1]。但考虑到将最大像素值1映射到网络的最大激活值，因为这可能导致某些激活函数（例如sigmoid函数）的饱和现象，使得梯度下降过程变得困难。因此乘以0.99是为了将像素值缩放到[0, 0.99]的范围内。又考虑到神经网络的权重初始化可能对零值比较敏感，因此加上0.01的像素值偏移。

**2. 搭建全链接神经网络**

定义Nerual\_Network类，类里包含激活函数，损失函数，前向传播，反向传播，训练模型和测试模型的函数。

初始化类函数：

输入层节点数：784 隐藏层结点数：200 输出层结点数：10

学习率（Learning grate）迭代次数（epoch）

初始化输入层与隐藏层的w(权重)和b(偏移):

输入层w(200,784) 输入层b(200,1) 输出层w(10,200) 输出层b(10,1)

**2.1 激活函数**

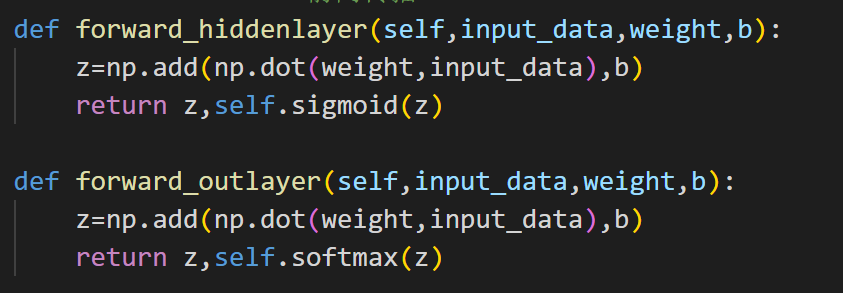
（1）隐藏层激活函数为sigmoid函数

求导：

（2）输出层激活函数为softmax函数

**2.2 前向传播**

原理公式：



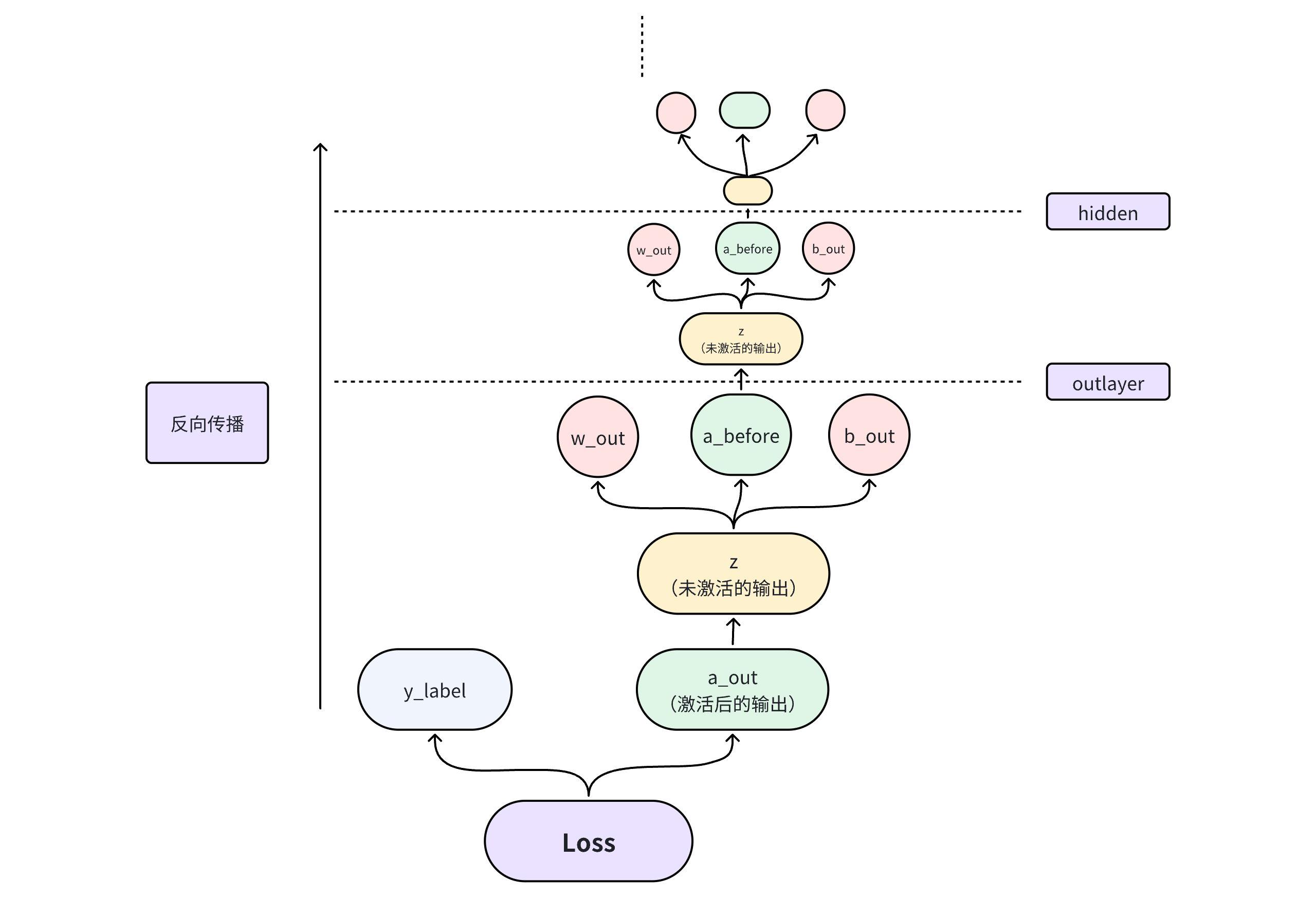
**2.3 反向传播**

（1）反向传播的基本原理

我们希望通过更新参数（w,b）使模型的损失达到最小。需要考虑w,b这两个参数对损失函数的影响有多大，可以用梯度来表示，也就是求Loss对w，b的偏导。

w梯度：

b梯度：



如图若想探究w对Loss的影响，可以先看w对z的影响，再看z对a\_out的影响，最后看a\_out对Loss的影响。

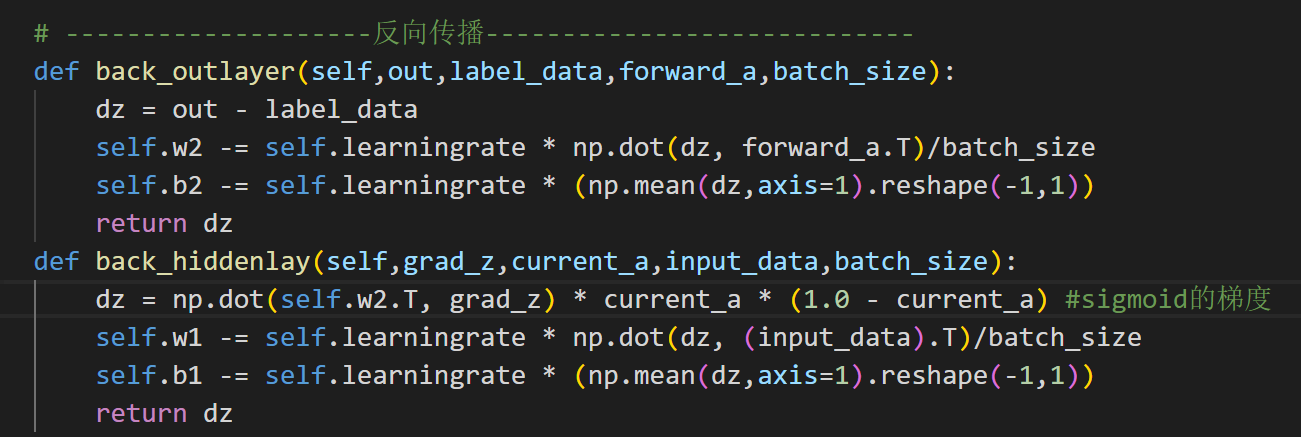
用数学方法求解，可以根据求导链式法则，求Loss对w的偏导，公式如下：

在本次实验中，为交叉熵和softmax激活函数求偏导化简后得:

为前一层的激活输出。

同理可求b的梯度：

隐藏层反向传播公式略有不同：



**2.4 损失函数**

**交叉熵损失函数**：交叉熵损失函数通过计算真实标签与模型预测之间的交叉熵，来度量模型的预测结果与真实标签之间的差异。

其中：

M——类别的数量

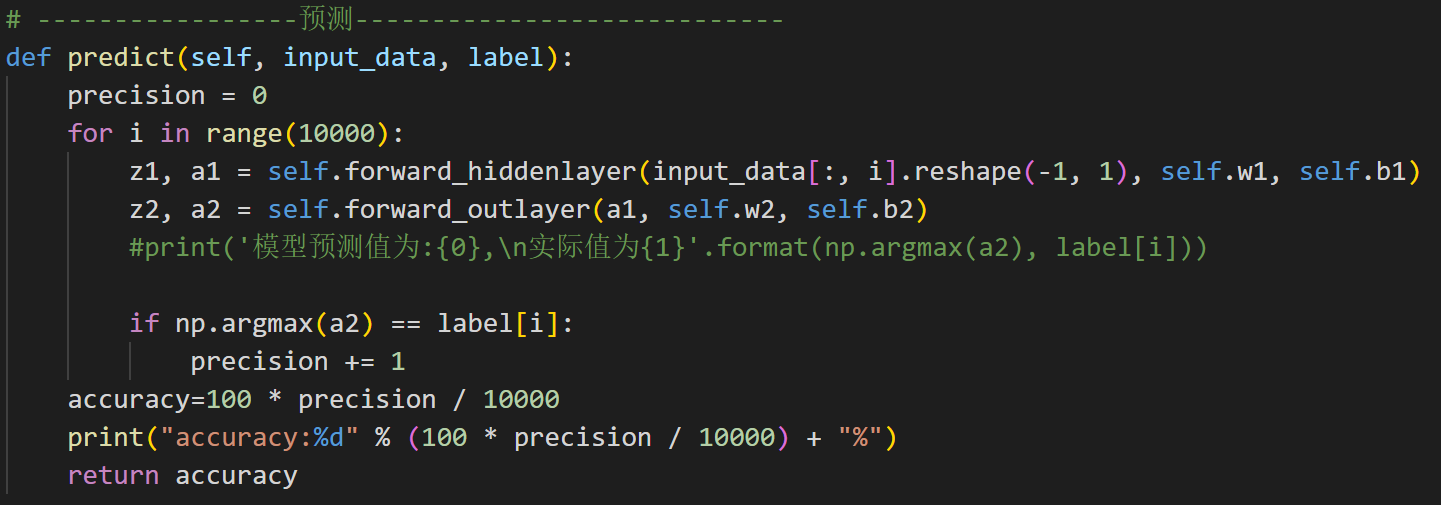
——符号函数（取0或1），如果样本i的真实类别等于c取1，否则取0

——观测样本i属于类别c的预测概率

**2.5 模型训练及预测**

（1）模型预测：

将测试集导入训练好的模型中进行测试，返回准确率。



（2）训练模型：控制训练轮次及迭代次数，调用以上定义的所有函数对模型参数进行更新。

**三、结果与分析**

**结果：**

运用mini-batch优化算法和L2正则化方法

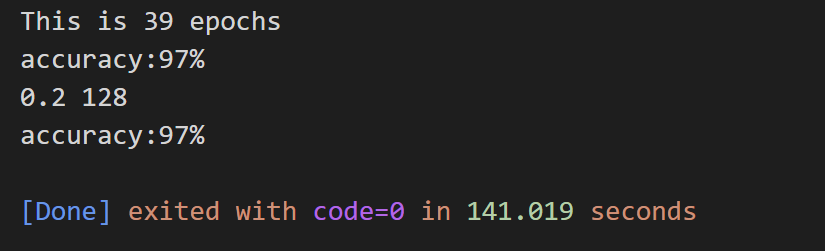
当batch-size=128 learninggrate=0.2 epoch=40 耗时140s，准确率可达**97%**

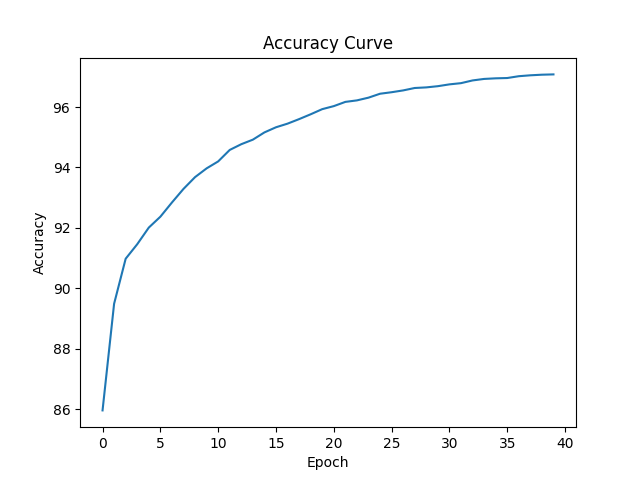
**调试过程**：

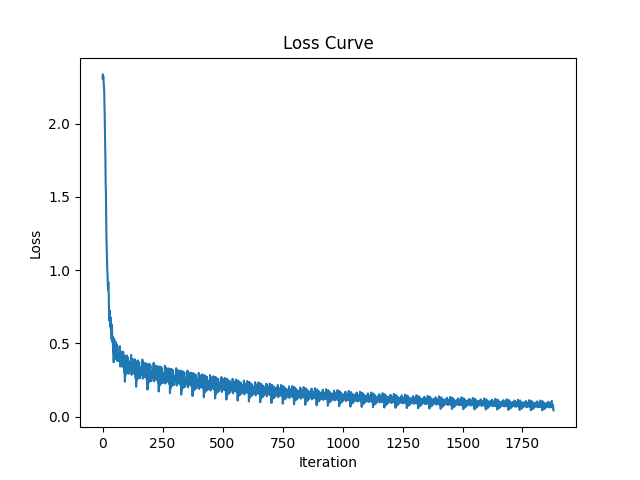
当batch-size=32或64，learninggrate=0.05时，学习效率大大降低，模型训练速度变慢，训练40个epoch准确率只有93%左右。尝试增加学习率为0.1，会出现softmax overflow现象。

当batch-size=256或更大 learninggrate=0.1 时，刚开始十轮收敛速度很快，到25epoch左右，模型似乎陷入了局部最小值，准确率不再提升，徘徊在95%不再提升。

当batch-size=128 learninggrate=0.05 时，需要训练80epoch左右才能收敛。

总而言之，通过控制变量法，发现batch-size=128 learninggrate=0.2 epoch=40是一组比较好的参数，耗时140s，准确率可达97%。





**分析：**

在使用优化算法前，我最先采用的逐个样本更新参数，学习效率低，准确率低且非常耗时。运行20轮需要20min左右，准确率只能达到90%。

使用mini-batch优化算法后，训练40轮后准确度提高到97%，耗时是之前的十分之一。小批量更新可以更有效地利用样本之间的相关性，减少参数更新的方差，从而加速收敛过程。

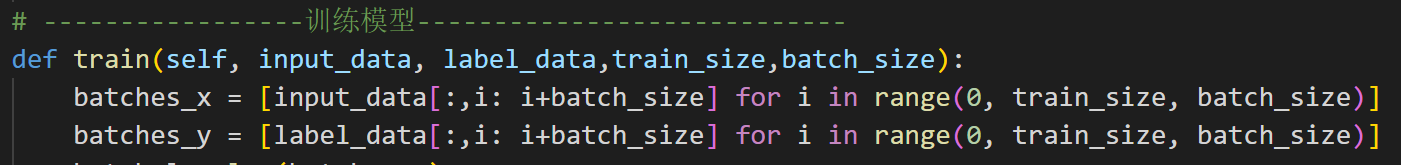
**四、拓展与创新**

**1. 优化算法**

*batch GD, mini-batch GD, SGD*

**1.1mini-batch GD**

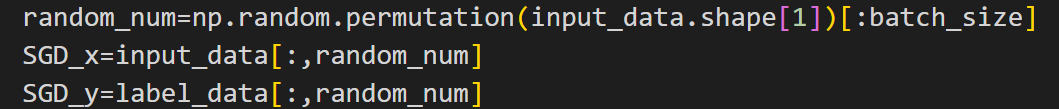
mini-batch梯度下降是一种将数据集划分为小批量样本进行参数更新的优化算法，相较于逐步更新参数的方法它具有更快的收敛速度。相较于full-batch，它大大减少了内存的需求，使得处理大规模数据集变得可行。



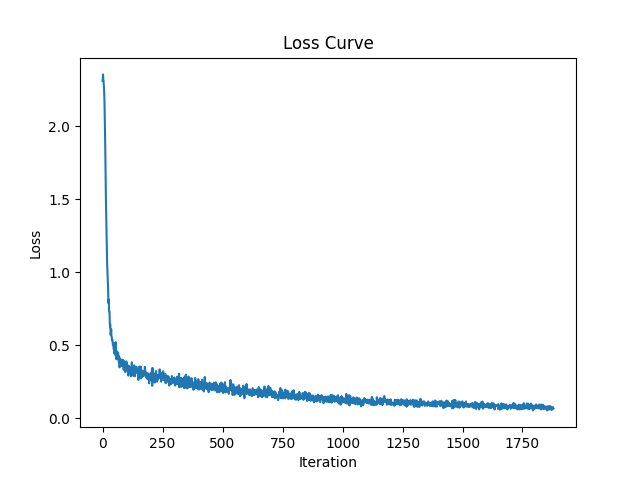
在训练模型之前将训练集数据和标签，按给定的batch-size划分成若干份，导入模型。与之前逐条训练更新参数不同的是，在反向传播中，对w,b参数的更新需要对w,b梯度进行平均操作。

**1.2 SGD**

随机梯度下降，随机选取一定批量的样本进行参数更新，它同样具有训练速度快，内存占用低等优点。相较于mini-batch，SGD每个小批量样本的选择是随机的，这样在每个迭代步骤中引入了一定的噪声，有助于避免陷入局部最优。



使用随机梯度下降需要注意的是，在随机选取样本时，要保证样本数据和标签是一一对应的。如上图，我使用random.permutation()函数生成一个随机排列的索引，让样本数据和标签都取这组随机排列索引就能保证数据和标签一一对应。



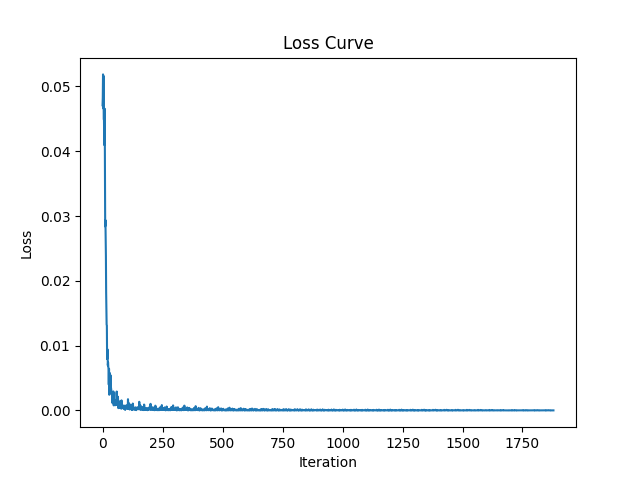
在相同batch-size,learninggrate,epoch下，对比mini-batch，学习效率提升了些，具体表现为前10轮SGD准确率比mini-batch GD高1%。但SGD的时间开销比mini-batch GD高，相同条件下，SGD训练时间是mini-batch GD的3倍左右。这可能是因为SGD每次迭代都需要随机选取样本，调用random.permutation()函数产生了较大的时间开销。

除此之外，我还注意到，使用SGD进行训练，每轮的迭代次数是自定义的，因此SGD实际上比mini-batch GD要多一个max Iteration参数，但在本次实验中我是先进行mini-batch GD优化，求得了样本可能被划分的块数，为了节省时间，我直接用mini-batch GD划分出来的块数作为SGD的max Iteration。

**2. 损失函数**

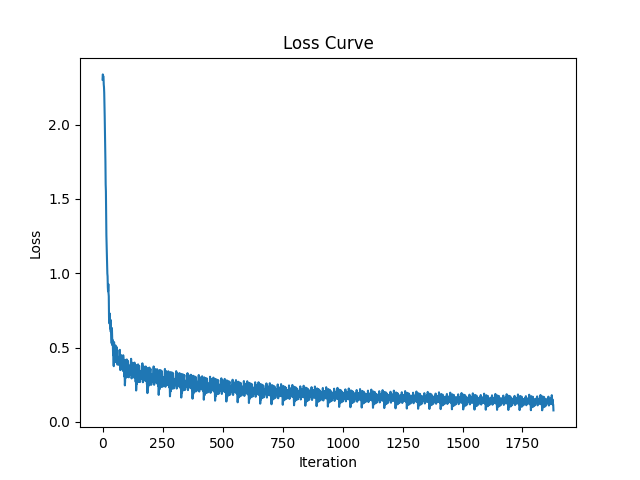
我尝试使用MSE作为该模型的损失函数，实验发现MSE损失函数比交叉熵更容易使LOSS达到最小值，这可能是因为MSE对预测值和真实值之间的差异进行平方，因此较大的差异会被放大，从而更容易导致损失函数的较大更新。

实验结果如下：



**3. 正则化**

尝试使用使用L2正则化进行模型优化。实验发现



L2正则化通过惩罚模型的参数值，使模型更加平滑，减少对训练集中个别样本的过度敏感。这样可以防止模型在训练集上过度拟合，即过度适应训练集中的噪声和细节，从而提高模型在未见过的数据上的泛化能力。

**五、遇到的问题**

**1. 数据处理**

问题：下载下来的数据为二进制格式，需要将之转化为numpy数组处理。

解决：因为MNIST数据集的二进制文件采用了大端字节顺序（big-endian）的数据类型，因此本实验中使用了字节顺序进行读取。使用struct.unpack()来解析每个文件中的头部信息和图像/标签数据，NumPy的 fromfile(train\_data,dtype=np.unit8) 函数将二进制代码解析为一个无符号整数。

**2. 参数优化**

参数不合适导致sigmoid激活后输出接近1，softmax激活溢出。

解决：调整batch-size大小，对于小批量，可以使用较小的学习率，以细粒度地更新权重；对于大批量，可以使用较大的学习率，以加快收敛速度。

当准确率提升很慢时，可以提高学习率。小批量学习时间长，但准确度高，大批量学习时间短，准确度相对低一些。

**六、总结**

1、图像识别流程及数据驱动方法：

图像识别的基本流程包括数据准备、模型训练和预测。在数据准备阶段，将手写数字图像转换为适合神经网络输入的格式。模型训练阶段使用训练集的图像和标签来训练神经网络，通过反向传播算法更新网络参数，使得网络能够学习到图像特征和对应的分类标签之间的关系。在预测阶段，我们使用训练好的模型对新的手写数字图像进行分类。

2、数据集划分和超参数调整：  
数据集通常被划分为训练集、验证集和测试集。训练集用于模型的训练，验证集用于调整模型的超参数，例如学习率、正则化参数等，以提高模型的性能和泛化能力。测试集用于评估模型在未见过的数据上的准确率。通过在验证集上进行超参数调整，可以选择最佳的超参数配置，避免过拟合或欠拟合的问题。

3、优化方法：

为了优化神经网络，可以使用不同的更新方法，例如随机梯度下降（SGD）和Adam等。这些方法通过调整权重和偏置来最小化损失函数，以便使网络的预测尽可能接近真实标签。

通过本实验，我们深入了解了图像识别的基本流程和数据驱动方法。我们实现了Softmax分类器和全连接神经网络分类器，并通过调整超参数和优化方法来提高分类性能。这些经验对于理解和应用深度学习在图像分类任务中的作用具有重要意义。未来，我们可以进一步探索其他神经网络架构和优化方法，以解决更复杂的图像分类问题。

**七、附录**

两个py文件：mini-batch.py和mini-batch\_regulation.py，其中mini-batch.py是小批量优化后的代码，mini-batch\_regulation.py 是L2正则化后的代码。