# Softmax实现手写数字识别

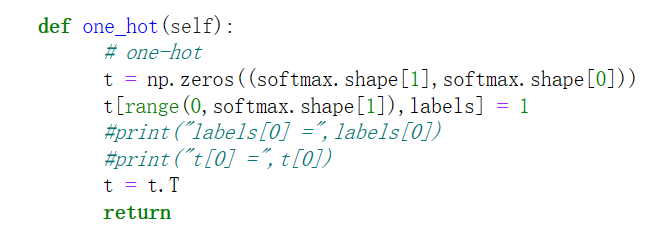
本次作业基于Softmax实现手写数字识别，根据作业一的要求完成前向计算和参数的更新，并通过调整超参数记录对分类性能的影响，此外在SGD中比较加入momentum和没有加入momentun对分类性能的影响。

在数据集方面，用的是MNIST数据集，包含60,000个训练样本和10,000个测试样本。MNIST中的数字范围是0到9，由此可知有10种类别（10种标签），分类数量K>2，更适合用K>2范畴Multinoulli/categorical分布的假设。

构造一个非线性的函数用描述样本x为k类的概率，该函数为softmax函数，在条件似然函数（独立性假设）的前提下，将最大化似然等同于最小化，即交叉熵误差函数。确认目标函数后，分别对W和b求偏导，计算梯度实现对参数W和b进行更新，上述为实现数字识别的基本思路。

## 数据处理方面

MNIST数据集，包含60,000个训练样本和10,000个测试样本。MNIST中的数字范围是0到9。由于考虑到一次更新成本函数的计算机开销比较大，因此将数据集进划为更小的集子，利用batch\_size参数为子集的样本数量，该方法用于随机梯度下降法中。数字范围是0到9，由此可知有10种不同的标签，需要将其转化为one-hot的表达方式。代码如下，创建一个全0的（n，10）的矩阵，其中n为样本数量，将下标与标签对应的值改为1。



## 前向传播

假设batch\_size为100，在forword函数中：

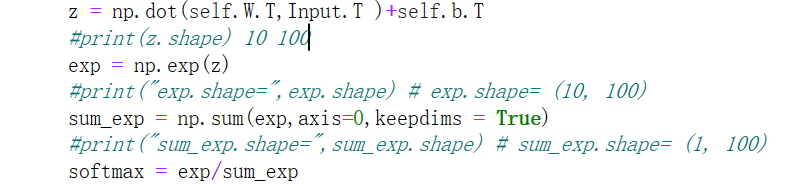
Input：为（784，100）的二维矩阵，表示有100个样本，每个样本有784个参数。

W：为（784，10）的二维矩阵，初始化为0的权重。

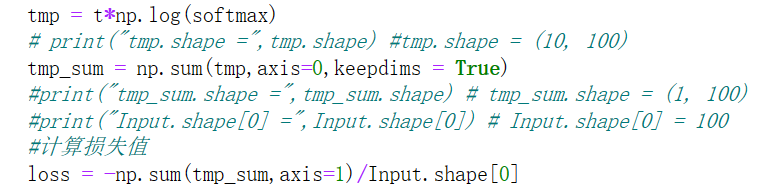
b：为（1，10）的一维矩阵，初始化为0的偏置。

经过矩阵相乘和元素积可以求出softmax函数，公式如下：

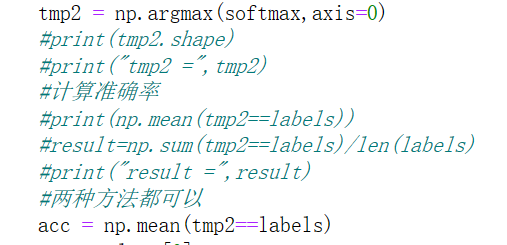




在求出softmax函数后，经过交叉熵误差函数可求出loss值。



再根据求出来的softmax函数求出accuracy，代码如下：



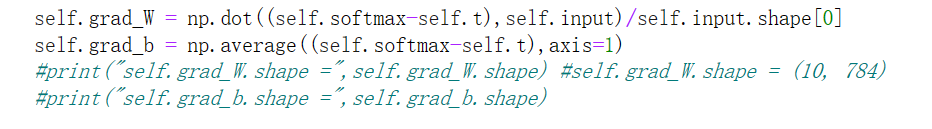
## 三、计算梯度

结合所求解出来的softmax函数的值，对W和b求偏导，公式如下：



其中f(x)为一个是(10，1)的矩阵，t为一个（10，1）的矩阵，x为（1，784）的矩阵，经过矩阵相乘得到是（10，784）的矩阵，共有N个（10，784）的矩阵进行求和操作，再除于100，可得到对W的偏导。

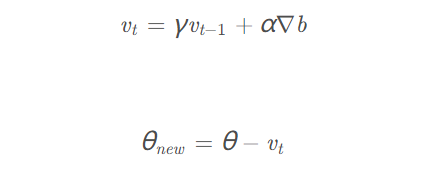
在代码中，seft.softmax是（10，100）的矩阵，self.t是（10，100）的矩阵，两者求和得到（10，100）矩阵，再与self.input是（100，784）的矩阵进行矩阵相乘，即完成了求和操作，直接除于样本100即可，代码如下：



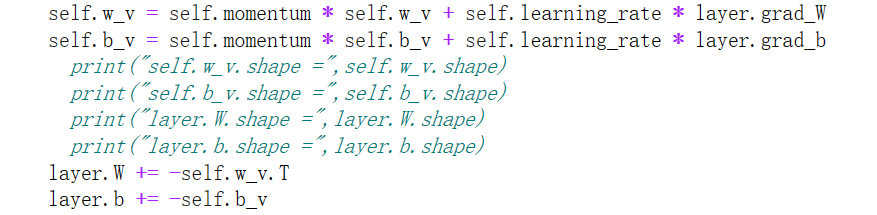
对b求偏导同理。

## 更新w和b

计算完梯度求出对W和对b的偏导后，即可更新W和b，这里引入了momentum动量法，公式如下：



更新代码如下：

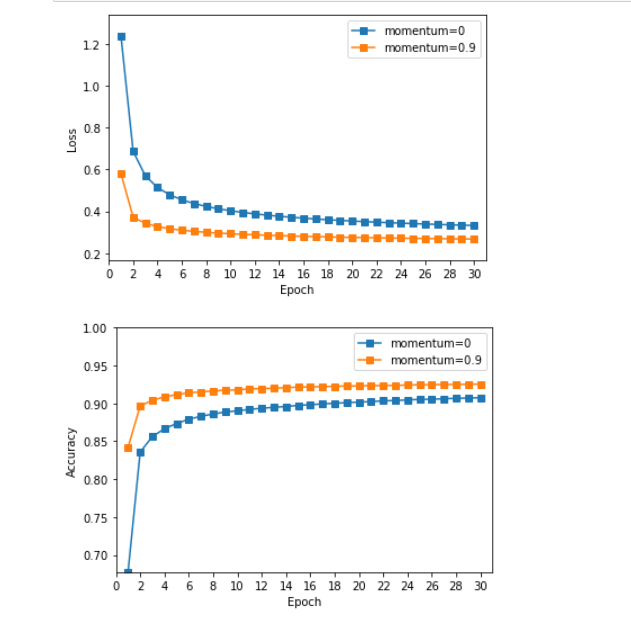


在实验中，对比使用momentum动量法和不使用momentum动量法的结果，一般将其设置为0.9和0。

## 五、实验结果

1.比较使用momentum和不使用momentum的实验结果：

其他参数为：'max\_epoch': 30,'batch\_size': 100,'learning\_rate': 0.01的情况下来进行对比，结果如图所示：

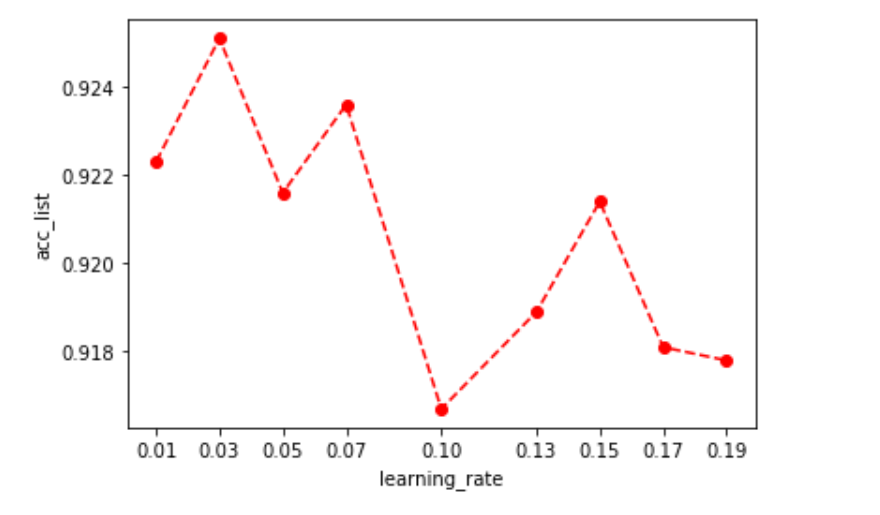


通过对比可以发现使用momentum后准确度大于没有使用momentum的准确度，而使用momentum后loss小于没有使用的loss。

两种方法训练时间分别为：40s和41s，在训练次数比较少时，并不差距不大，如果将训练次数加到100次，两种方法的时间分别为141s和144s，相差了3s。momentum 算法思想：参数更新时在一定程度上保留之前更新的方向，同时又利用当前batch的梯度微调最终的更新方向，简言之就是通过积累之前的动量来加速当前的梯度，因此使用momentum后，收敛比较快，所需要的时间比较短。

1. 调整学习率观察对准确度的影响。

其他参数为：'max\_epoch': 30,'batch\_size': 100,'momentum': 0.9的条件下。

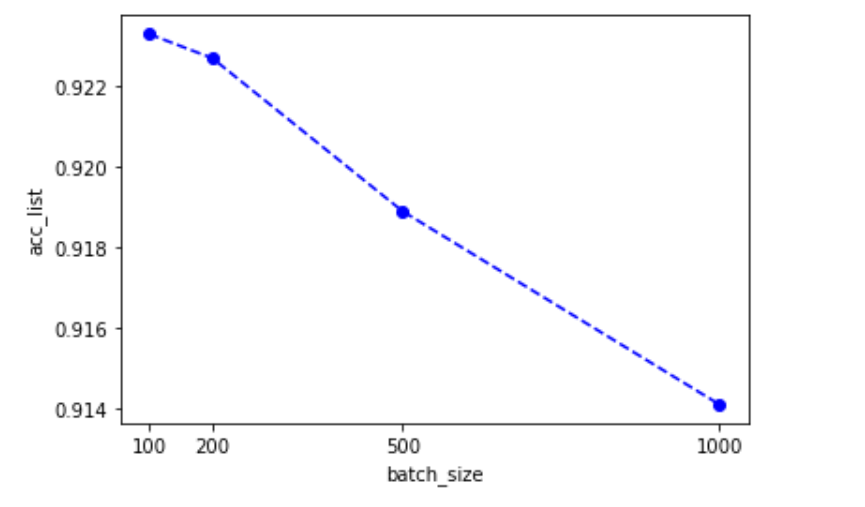


由此可见，并不是学习率越大越好，虽然学习率增大会加速梯度下降的速度，但可能错过最优解，图中学习率为0.03，准确度最高。

1. 调整batch\_size观察对准确度的影响。

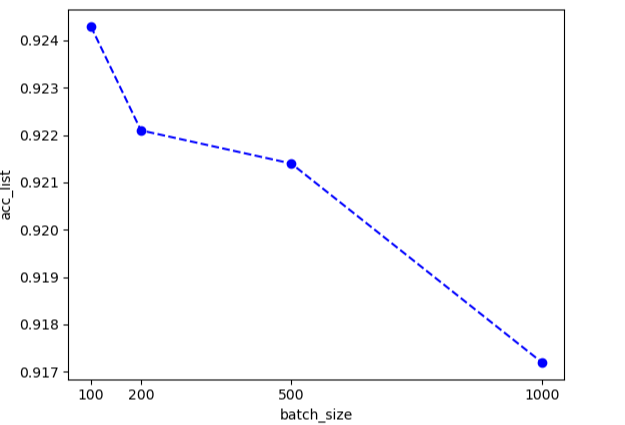
其他参数为：'max\_epoch': 30,'learning\_rate': 0.01,'momentum': 0.9的条件下。

batch\_size分别为100，200，500，1000，结果如下图所示：



有图可知，batch\_size越大，准确度越低。

将调整'max\_epoch': 50，如图所示：



虽然增大max\_epoch的数量，但是准确度依然递减。通过对比两图，可以发现增大max\_epoch的数量，4个测试点的准确度都有所提升。