softmax\_regression

# 实验要求

在这个练习中，需要训练⼀个分类器来完成对MNIST数据集中 0-9 10个手写数字的分类。

初始代码可以在课堂提供的压缩包的 ex1/ 中找到。

MNIST数据集可在数据集下载地址中下载，共有4个.gz文件，解压后放在初始代码的 ex1/ 目录下。

初始代码中已经包含以下内容：

• 在 main.py文件中加载训练和测试数据集

• 在 main.py文件中调用训练和测试相关函数

• 在 train.py文件中调用 softmax\_regression() 函数

当训练完成，分类器会输出10个类别的测试准确率。

你的任务是实现 softmax\_regression() 函数，计算每一次迭代的损失值 J (θ, x, y)，将它存储在变量 f 中，并计算梯度 ∇θ J (θ, x, y)，将它存储在变量g中。初始代码会将 θ 的形状定义为⼀个 k×n 的矩阵 (K=10个类别)。

此外，你需要在 evaluate.py文件中实现cal\_accuracy()函数，输出分类器在测试集上的准确率。

⼀开始你可以使用 for循环 来得到正确的梯度 (最好尽早使用调试策略来进行梯度检查)。之后你会发现用 for循环 实现的代码计算速度太慢了。当你得到正确的梯度之后，尽可能尝试对你的代码

进行优化，再用完整的数据集来跑实验。

程序中涉及到的超参可以自行设置，尽可能得到较高的分类准确率。

# 实验内容

## Softmax\_regression()

在本数据集中，k=10，m=60000，n=784。

Input：

* theta：参数矩阵，k\*n矩阵，初始化为随机赋值且每行和为1
* x：输入数据的特征矩阵，m\*n矩阵
* y：样本特征，利用独热编码的方式初始化，其方法是使用N位状态寄存器来对N个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候，其中只有一位有效。也就是说如果y[i,j]=1，就以为着第j个样本特征为i。
* iters：训练轮数
* alpha：学习率

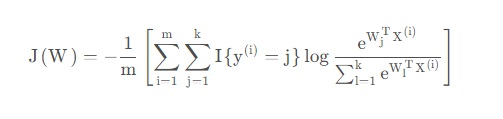
Output：

* theta



### 计算损失值：

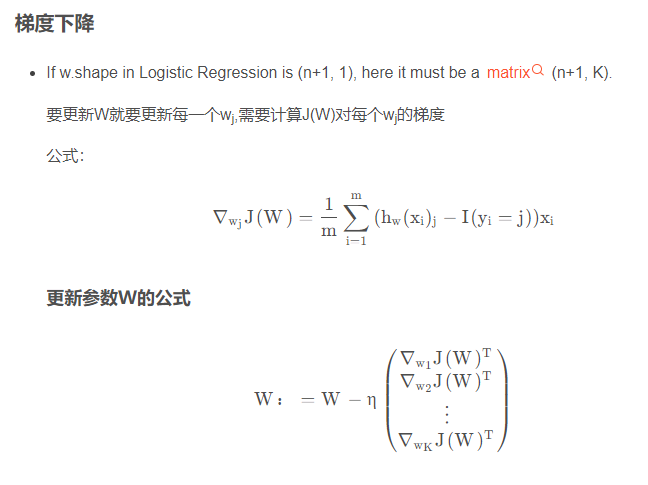
将矩阵x和矩阵theta相乘，得到每个样本特征为0~9的概率；



将样本特征概率指数化，并标准化（使得e^x后，每一行之和为1）；

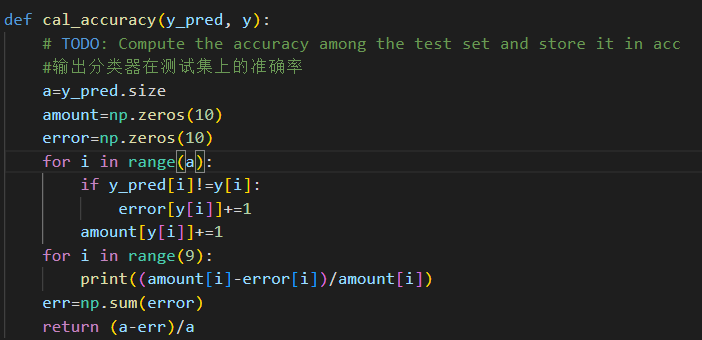
YT[i,j]=1意味着y(i)=j为真，即I{y(i)=j}=1，所以损失值只需减去YT[i,j]=1对应的log(probs[i,j])之和。

### 计算梯度



根据上图所示公式写的，得到的矩阵应和theta同为k\*n矩阵

## Cal\_accuracy()

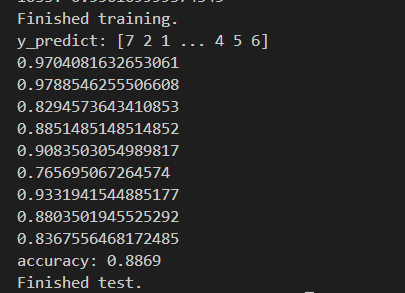


将预测集和实际的结果一一比较，对每一个特征求出正确率。

正确率公式为：（特征i的总个数-预测错误的个数）/特征i的总个数

并求出对整个数据集的正确率。

# 实验结果



轮数为1000，学习率为0.1

输出：

每一轮的损失值（以此确定程序是正常运行的）

每个样本的预测特征

从0~9的正确率

总的正确率