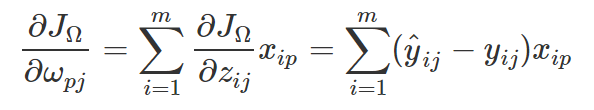
Softmax回归模型（多元分类）

初期

一开始主要在模型的构建上花费时间。了解了一些神经网络的概念后，我决定使用只有输入层和输出层的简单的神经网络结构来构建模型，其输出层激活用softmax，共有784个输入层结点，10个输出层结点。

在这个过程中，我首先面对的第一个难点是矩阵的运算。一开始我只是简单地把权重W初始为（10，784）的矩阵，然后把单个样本的特征值X化为一个（1，784）的矩阵进行W\*X.T+b的公式运算，但是在处理样本量比较大的情况下用for循环遍历全部样本就使得效率十分缓慢。后来想到了直接将全部样本数据化成一个（2w，784）的矩阵直接进行W\*X.T+b运算,既提高了效率，又简化了代码量。

我面对的第二个难点则在用梯度下降更新权重时求偏导的问题上。查阅了网上很多博客，但我始终对偏导的形式不甚理解。在跳过反向传播公式的推导，我直接采用了以下的公式求偏导：

其中，i是样本个数，j代表第几个输出结点，p代表第几个输入结点。

在使用梯度下降的过程中，我学习了不少的东西。比如使用交叉熵作为模型的损失函数时，我了解了一些关于信息熵，条件熵和信息增益的相关概念和公式，这使得我进一步去了解了决策树和随机森林的一些大致原理。这些东西虽然还没来得及用代码去实现使用，但是这种学习过程让我印象深刻。

中期

在模型构建完载入测试集进行数据预测后，我发现分数不太理想，于是在这阶段中我主要花时间在对数据样本的初始化处理，参数的调整以及思考探索可以用什么方法提高模型预测的准确度。

在数据样本的预处理上，我先在训练集上找出值全为0的特征进行删除，删除了大概九十多个特征值，并在测试集上也把这九十多个特征值进行删除，然而结果并没有变化多少。而且后来通过查找索引发现，这九十多个特征有一些在测试集上它的值并不全为0，我并未搞清楚如何处理这种情况，便放弃了这种删除特征值的方法。

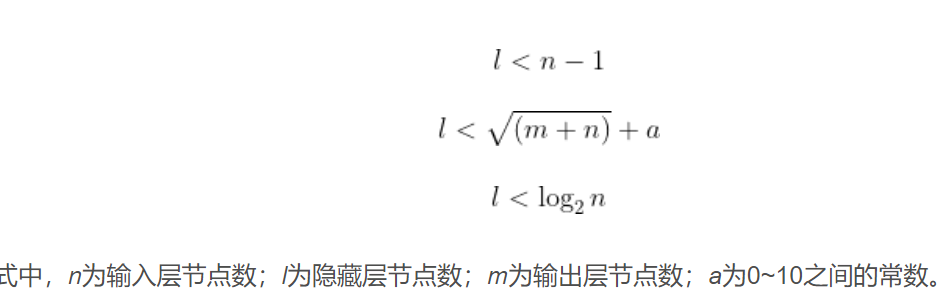
在参数的调整上，我设置梯度下降的迭代次数一万次，然后不断调整学习率获得较少的损失值，发现了学习率在0.3时，损失值普遍较少。而目前我最高的分数也是在没有对数据进行预处理的情况下使用这个学习率得到的。

由于测试集只给出了特征量的数据，难以知道模型预测出来的结果与实际结果相差多大。在样本量有两万的情况下我便决定将15000样本划分为训练集，5000样本作为测试集，并将模型预测出来的结果与测试集的实际结果进行损失值的计算。结果发现，测试集的损失值有0.3几，而训练集的损失值只有0.1几。这让我思考是不是模型存在过拟合的问题，我便在模型里添加了正则化处理，对除偏置外的所有权重进行惩罚，在尝试了一些正则化参数lamda值后发现模型的分数并未提高，甚至有所下降，便放弃了这个处理。

总的来说，这个过程我不断进行模型优化、评估，来提高模型预测的准确度，但是效果都不太理想。其中，我学习了解到的一些诸如PCA的特征处理方法由于时间原因尚未进行实际运用。

后期

由于一系列模型优化的尝试都没有获得较大的提高，我便开始着重于提高模型的复杂度以此进行这类非线性的分类问题。在加深了对神经网络知识了解后，我开始考虑增加一层隐藏层。查阅博客发现Relu激活函数在大多数的情况下比sigmoid函数表现要好，我便使用Relu作为隐藏层的激活函数。紧接着，隐藏层里结点个数的选择又是一个问题，我在查阅博客时发现了三个关于结点个数确定的公式：



我便以此确定隐藏层的结点数为5个。增加一个隐藏层在模型原有的基础上改动并不是很大，只是添加了一个Relu函数和重新设置了从输入层到隐藏层，隐藏层到输出层的两组权重参数

多了一个隐藏层的关键难点在于权重的更新。由于Relu函数的形式，在对第一组权重梯度下降求偏导时，当输入值大于0时，该梯度与后一组权重梯度相同；当输入值小于等于0时，梯度则为0。

相关效果还在尝试中……