神经网络作业一实验报告

191250026 丁云翔

主要工作

使用了tensorflow框架，参考了官方手册和教程进行实现，搭建了一个多层感知机对MNIST数据集进行分类。main.py是入口函数，负责读取MNIST数据、调用各项学习方法、并绘制最后的性能图。MLP相关代码位于baseline.py，实现了一个单隐层的多层感知机，采用sigmoid激活函数，训练轮数为50轮。其余文件均为baseline的变体，修改了对应的需要对比的点。

如何运行

运行main.py即可。

各项方法对比

不同的参数初始化方法

这里选择了全0初始化来与baseline方法的随机初始化对比。

不同的损失函数

这里选择了均方误差来与baseline方法的交叉熵损失对比。

不同的学习率调整方法

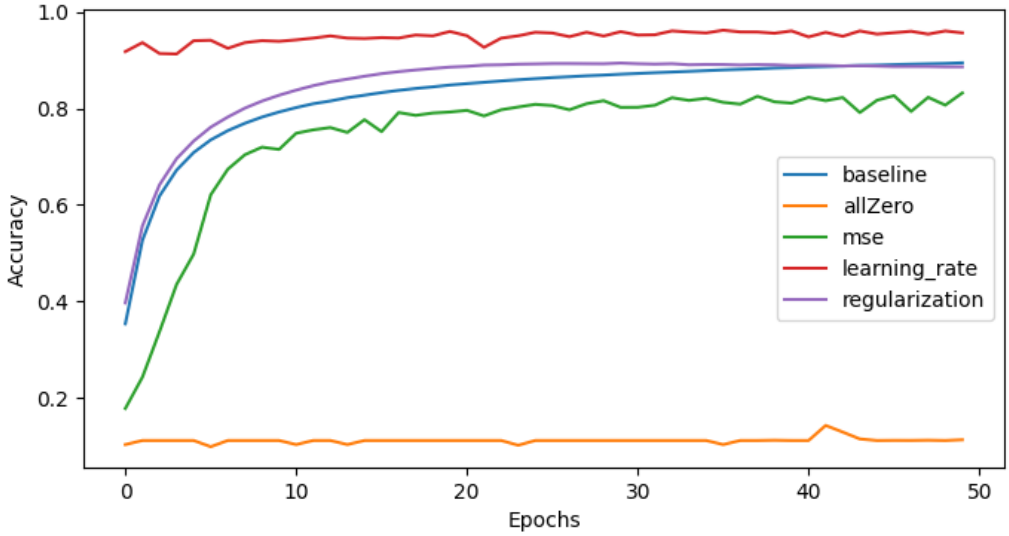
这里选择了adam自适应学习率方法来与baseline方法的固定学习率对比。

不同的正则化方法

这里选择了L2正则化方法（正则化项权重设为0.001）来与baseline方法的无正则化对比。

对比结果

方法	50轮后的测试集准确率	训练集准确率
baseline	0.8944	0.8935
全0初始化	0.1138	0.1162
均方误差	0.8320	0.8413
adam自适应学习率	0.9564	0.9502
L2正则化方法	0.8861	0.8937



全0初始化让MLP无法训练，故准确率一直保持在11%左右；均方误差性能不如交叉熵，因为是多分类而不是回归问题，使用交叉熵损失函数会更合适一些；采用自适应学习率比固定学习率更灵活高效，故收敛也更快，且性能更好，可以找到全局最优；使用L2正则化后训练集准确率提升更快，但最后达到收敛时训练集准确率比不正则化要略低一些，可能是因为防止了过拟合，并且使用L2正则化后测试集准确率略有提高。

最后选用的方法

随机初始化，交叉熵损失函数，adam自适应学习率方法和无正则化。

因为adam自适应学习率方法和L2正则化都能提升性能，但我测试了一下同时采用这两种方法，性能反而下降了。所以还是只改进成adam自适应学习率方法性能最好。50轮内训练集准确率最高为0.9604，最终测试集准确率为0.9502。