构建两层神经网络分类器

实验报告

**胡诚驿 19300180073**

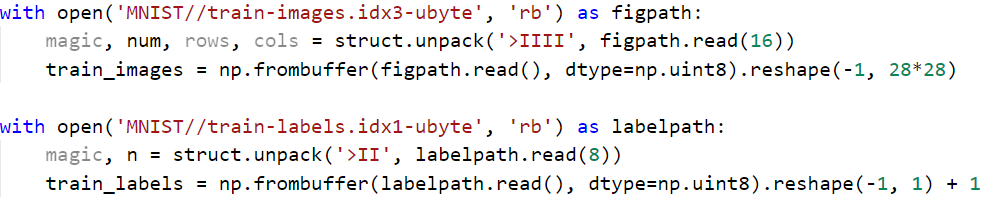
数据概况：

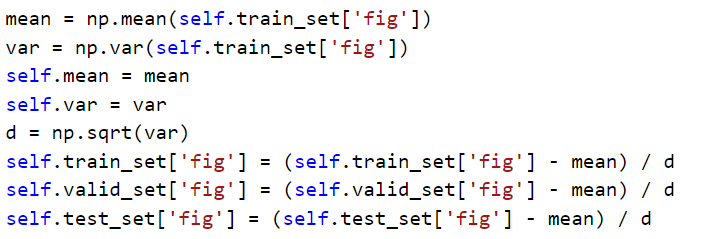
MNIST数据集包括了60000个手写字体相关的训练集以及10000个测试集样本，单个图像的大小为28 × 28 = 784。

代码组成：

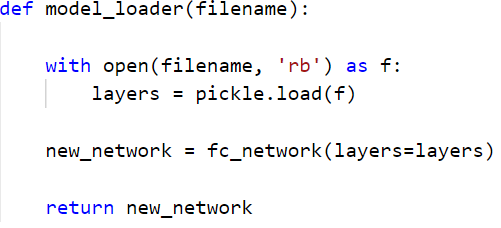
**data\_load:** 加载来自MNIST数据集的数据，划分为训练集、测试集与真实数据集，同时对图像数据进行标准化处理，并定义函数以实现单个数据的提取与图像的可视化。

为了在不使用GPU的情况下较为高效地使用原数据进行训练，经过处理我们得到了10000个图像作为新的训练集数据train\_set，将其余原训练集归入valid\_set，同时保留原测试集为test\_set，由此得到三个字典类型的数据集用于下一步的测试。

训练集导入。

数据标准化。

**model\_load：**基于训练结果与给定的文件名称加载并保存相应的 .tkl格式的模型文件。



**network：**该神经网络的主体部分。方便起见我们将对数损失函数同样保存在该部分代码中。此外，我们定义的类fc\_network包含了train、predict、model\_save三个部分，分别实现对应模型的训练、预测（同时得到每批数据的损失与错误率）与模型的保存。

此外部分参数信息如模型损失与测试精度曲线的绘制及图像保存同样在这一部分实现。

**layer：**该神经网络各层的具体内容构建，便于network部分代码引用以得到网络。

此处我们引用了神经网络与深度学习课程中使用的内容，为代码简洁起见此处只包含了全连接层、激活层与SoftMax层的代码，

**train：**针对不同的学习率，隐藏层大小，正则化强度集合进行同一结构神经网络的训练（即包含了参数查找部分），并根据平均损失与精确的等结果的比较判别确定最终的参数。

训练：

该部分主要实现于代码文件network与layer部分。具体实现过程与分析如下：

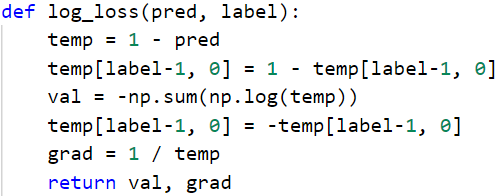
激活函数：

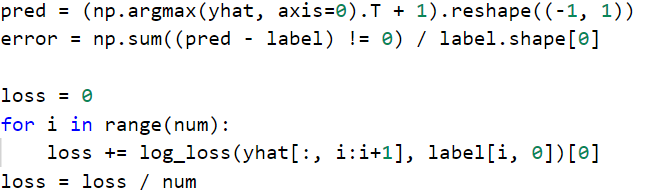
在确定的网络结构（两层神经网络）与后续参数查找工作保障前提下我们优先选择ReLu函数作为该网络的激活函数，相比于Sigmoid等激活函数它能够更好地避免梯度消失等问题而且具有相对较快的收敛速度。

我们在layer部分的代码中保存了Sigmoid、ReLU、tanh等激活函数的前向与后向处理，以便于后续可能的修改需求。

loss：

我们在此使用对数损失函数获取loss。相关代码：

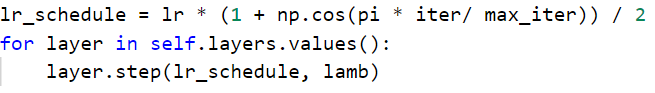




反向传播&随机梯度：

**layers**中的每一层引入反向传播函数更新参数。

学习率下降策略：

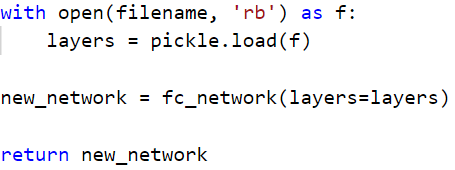


偏置项：

在该神经网络模型训练试验中我们默认引入偏置项，提高拟合能力与速度。

保存模型：

此处为了保存训练出来的模型我们另外建立一个model\_load 文件实现所得模型的简易保存，每次训练都在最后导入储存对应的模型，最终我们只保留试验得到的最佳模型。



参数查找：

我们使用类似网格搜索的方式对三个参数即学习率、隐藏层大小与正则化强度等几个重要参数，选取若干个典型取值对神经网络进行训练，并根据结果初步选定训练效果最佳的一批参数作为最终结果。各参数的具体实验情况如下：

学习率：

我们选取了[1e-3，5e-4，1e-4]三个学习率取值投入到模型的训练当中。在相同条件下1e-4大小的学习率在300000次迭代后仅取得约6.1%的学习效率，相比之下5e-4学习率取得的2.94%错误率与1e-3学习率取得的2.34%错误率显然不足。最终选择1e-3（0.001）作为该神经网络模型的学习率。

隐藏层大小：

我们选取了[100，300，500]三个神经元数量取值投入到模型的训练当中。在相同的学习率等条件下该集合取得了对应[2.95%，2.34%，2.39%]的错误率结果，loss情况与此类似。因此为了同时确保模型精度与运行效率，我们令该模型的隐藏层大小为300。

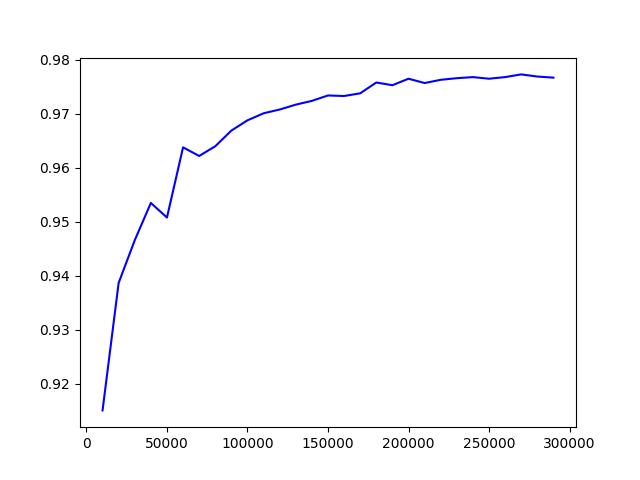
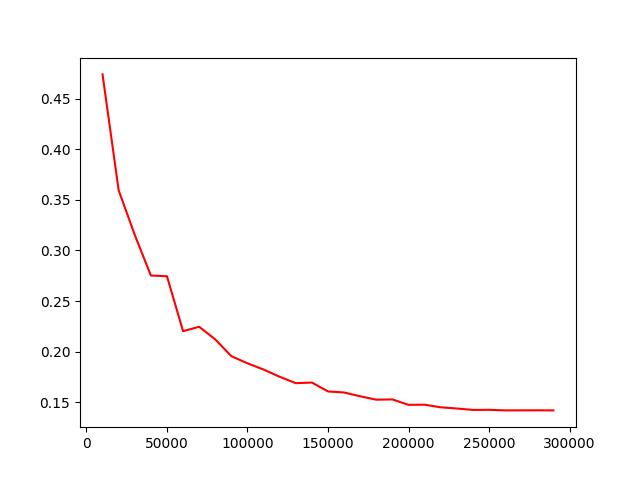
正则化强度：

我们选取了[0.1，0.01，0.001]三个λ取值投入到模型的训练当中。对应地在前两项最佳参数的条件下取得了[9.0%，3.2%，2.4%]的错误率结果，可以初步认为相同条件下更小的取值能够带来更好地训练效果。因此我们直接选取0.0001作为我们最终的λ取值。

测试：导入模型，用经过参数查找后的模型进行测试，输出分类精度：

结合**train**部分的代码与前述步骤我们实现了不同参数下不同模型的训练、测试并保存相关模型，最终我们采取初步判定的最佳参数代入，得到了用于可视化与最终提交的模型。

相关曲线绘制：

蓝色曲线为精确度随迭代次数变化的记录，对应的红色曲线为损失量的记录。

Github Repo: https://github.com/IdeoHu/HuChengyi-cv\_pj1

DBCloud: http://210.16.187.147:40888/share/Kk56HFQq?diskhost=http://210.16.187.147:26587