两层神经网络分类器—以 MNIST 数据集为例

向泉州 22210980116

摘要

这是《神经网络与机器学习》的第一个项目。在这个项目中，我们基本上用 Python 和 Numpy 建立了一个多层感知机。我们训练出了一个有效神经网络，得出了训练和测试的损失以及误差并将其可视化。此外，我们刻画了 weight 和 bias 的分布，还用降维技术将每层的参数可视化。

一、项目总览

本项目的实现试图模仿 Pytorch 的架构，它的可读性更强且更好用，但在可能会让读者没法真正理解神经网络的架构及为何实现。整个软件包被列在 ./module 目录下。代码可以在 https://github.com/AltonXiang/MNIST，训练后的模型被列在 ./model 目录下，可以直接使用。

本项目包含了以下文件：

parameter searching.ipynb：寻找模型的最优参数，展现 loss 与 acc，并把测试结果最好的模型保存

test.ipynb：测试训练后的模型

visualizer.py：可视化参数

module:

- data.py：读取 MNIST 数据集
- dataloader.py：构造了一个 dataloader 函数
- function.py：提供本项目所需的所有损失函数、激活函数和线性层。
- nn.py：定义了一个 Module 类来刻画我们的神经网络
- optim.py：构造了一个 SGD 优化器
- train.py：定义了训练的整体步骤

model:

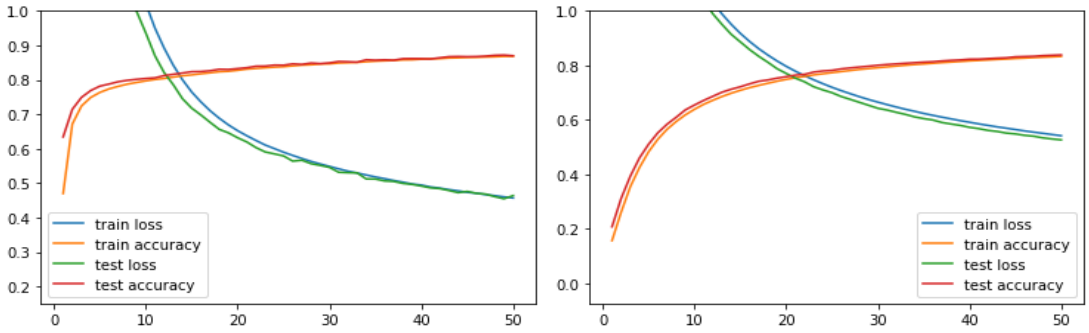
- mnist_model.pkl：我们训练好的神经网络模型

data：下载的 MNIST 数据集

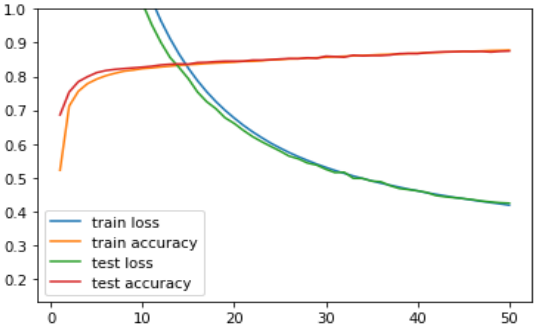
二、参数查找

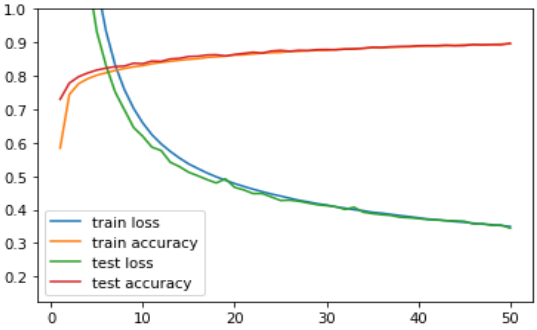
我们对模型的关键参数进行了查找与比对，包括激活函数、学习率、隐藏层大小、权重衰减（正则化强度）。结果如下：

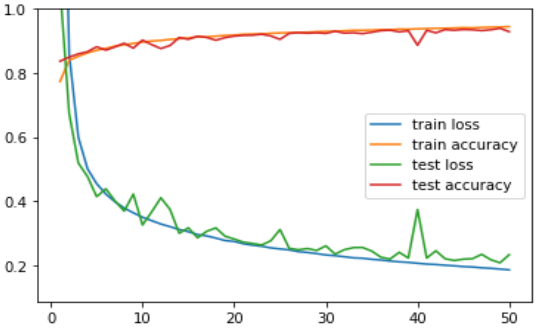
（1）激活函数：选择 Relu 还是 Sigmoid？（结果显示选择 Relu）



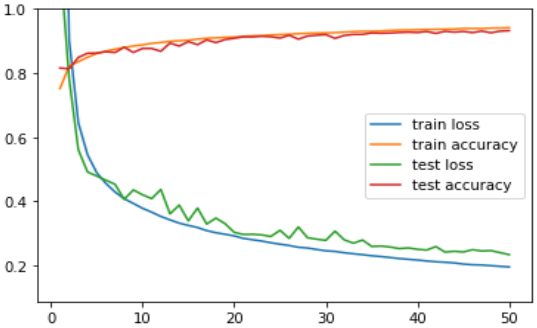
（2）学习率：0.05 还是 0.1 还是 0.5？（结果显示选择 0.5)

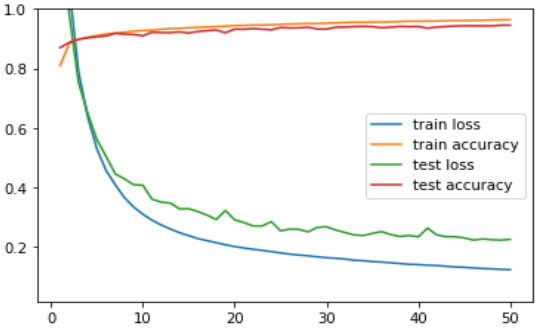


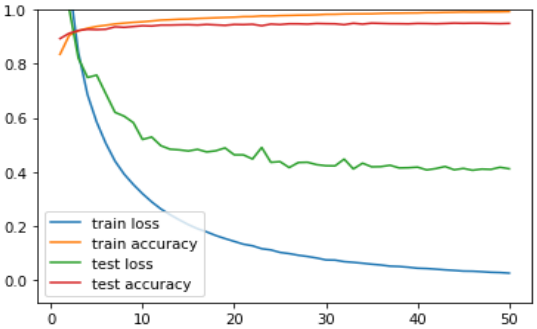




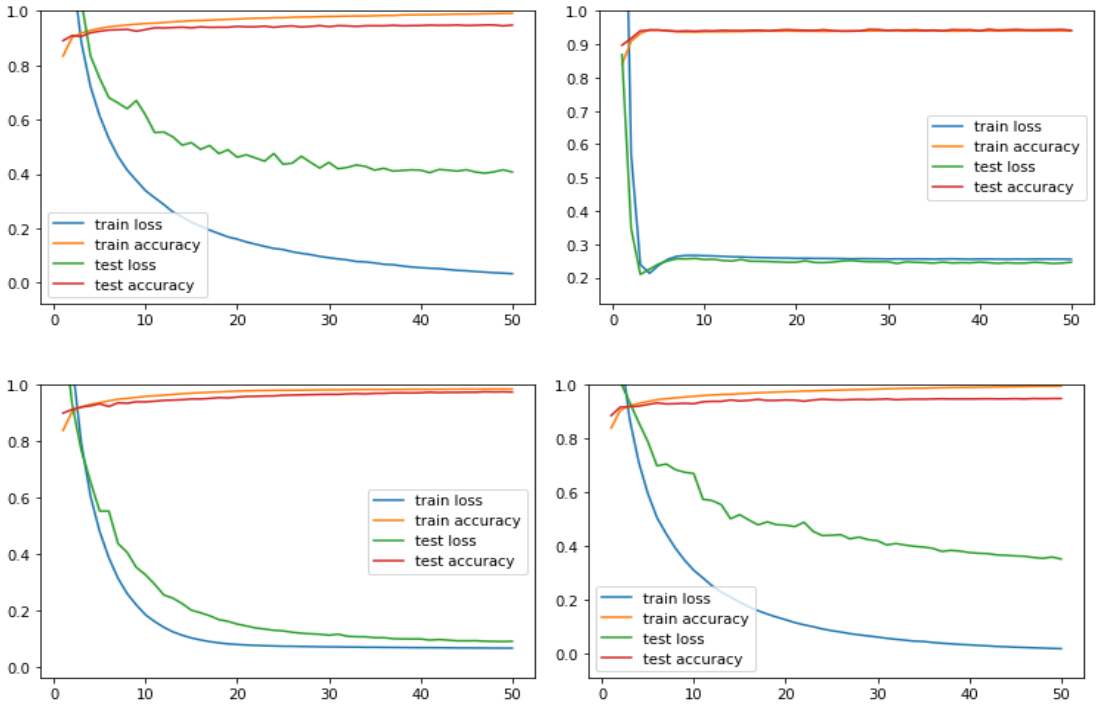
（3）神经网络层数：64 层还是 128 层还是 256 层？（结果显示选择 256 层）







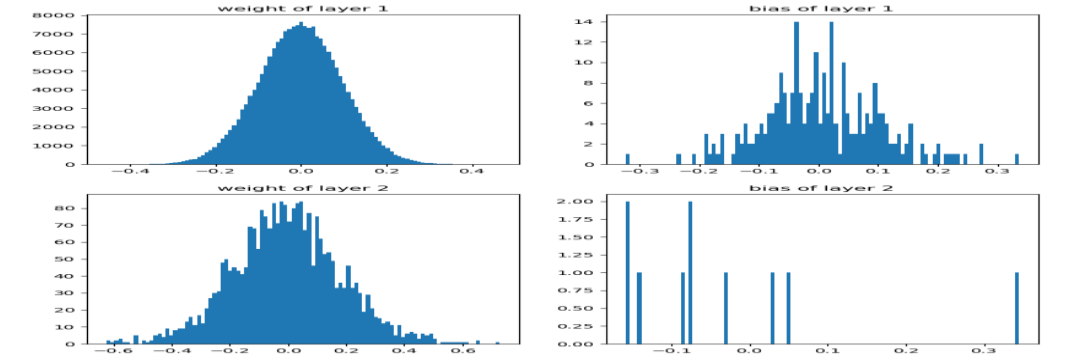
（4）权重衰减：None 还是 1e-3 还是 1e-4 还是 1e-5？（结果显示选择 1e-4）

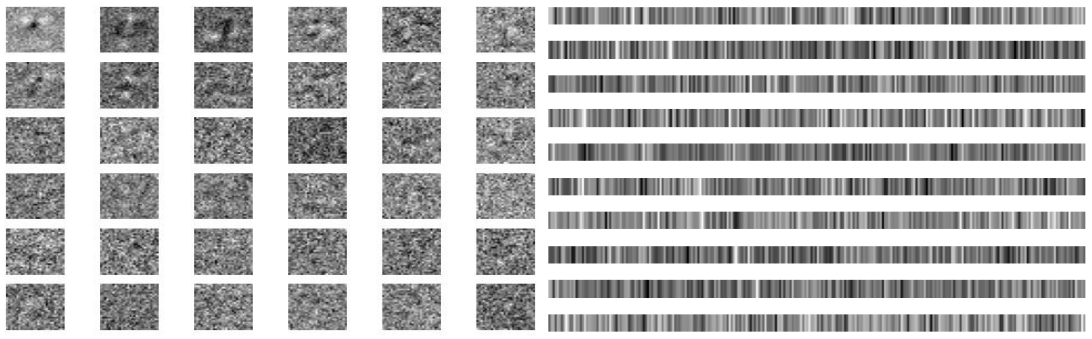


因此，我们最终选择的模型为[Linear(784, 256), ReLU(), Linear(256, 10), Softmax()]，学习率为 0.5，权重衰减为 1e-4。将训练好的模型传入 test.ipynb 进行测试，我们得到的 loss 和 accuracy 分别为 0.0866 与 0.9748。

三、可视化

我们首先可视化 weight 和 bias 的分布，随后分别将两层的参数可视化出来。由于第一层的权重数量较大，我们采用主成分分析的技术将其进行提取，并成功可视化。





Layer1-weight　　Layer2-weight

Layer1-bias

Layer2-bias