,

实验报告

用 Numpy 完成两层神经网络

姓名：孟博宇

学号：21210980114

授课教师：张力

课程：神经网络与深度学习

目 录

1 实验要求

构建两层全连接神经网络模型，对 MNIST 数据集进行分类，要求包括激活函数、反向传播、学习率下降策略、L2 正则化、SGD 优化器以及模型的保存和载入。对不同的学习率、隐藏层大小以及正则化强度做对照实验，最后上传自己的保存模型。

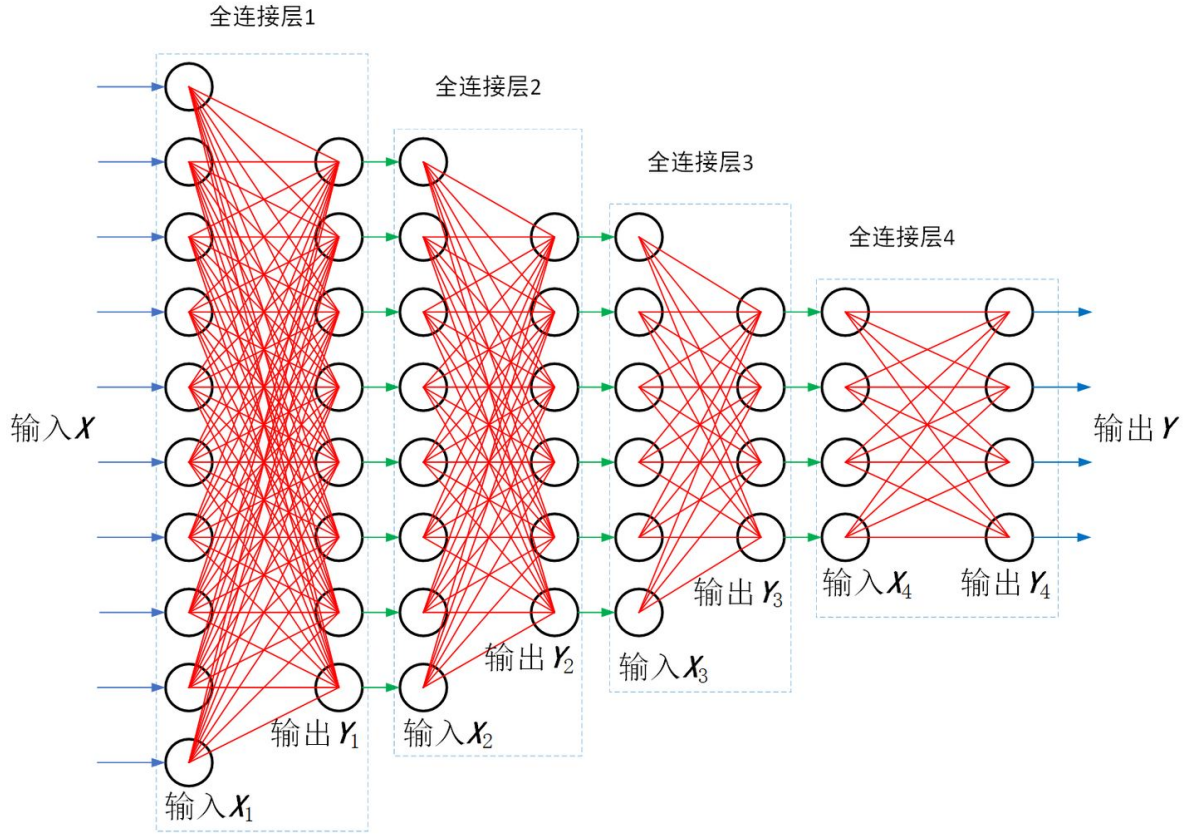

图 1-1 神经网络示例图

2 实验原理

2.1 神经网络

传统的感知器和线性分类器都有自身无法克服的缺陷，它们都不能解决线性不可分的问题，因此在实际应用过程中受到了限制。而神经网络具有很好的泛化性和非线性映射能力。

BP 算法是神经网络的核心，其基本思想是：把学习的过程分为两个阶段：第一个阶段是信号的正向传播过程——信息通过输入层、隐藏层、激活层等逐层处理，并计算每个单元的实际输出值，最后输出结果与真实值计算损失值；第二个阶段是反向传播过程——利用输出值与真实值计算每一层的梯度，根据梯度利用优化算法调节每一层参数的数值。整个过程不断迭代，最终得到训练好的网络，输出我们期望的结果。

2.1.1 问题定义

假设一次选择 p 个样本，每个样本的维度为 m，我们可以将这批样本用矩阵 X 来表示，其维度为 $p \times m$。每一层的权值矩阵 W 的维度为 $m \times n$。输出向量 Y 维度为 $p \times n$，以及偏置向量 b 的维度为 $1 \times n$。

2.1.2 损失函数

损失函数用以监督训练进展，确保我们正在向正确的方向移动。通常来说，损失函数是由真实值和预测值来确定的，损失函数的输出值越低代表我们的训练越有效。在分类问题中，我们通常使用交叉熵来确定损失值。首先我们使用 softmax 函数将输出值变为概率：

$$p = softmax(x_i) = \frac{e^{x_i}}{\Sigma_k e^{x_k}}$$

$$L = Loss(p, y) = -\sum y_i \log(p_i)$$

其导数为

$$\begin{aligned}\frac{\partial L}{\partial x_i} &= \sum \frac{\partial L}{\partial y_j}\frac{\partial y_j}{\partial x_i} \\ &= \sum_{i \neq j} y_j p_i + y_i(1 - p_j) \\ &= y_i - p_i\end{aligned}$$

2.1.3 线性层

线性层类似于线性回归，其公式为

$$Y = XW + 1 \times b$$

其难点在于反向传播的过程

$$\nabla_X L = \nabla_Y L \times W^T$$

$$\nabla_W L = X \times \nabla_Y L$$

$$\nabla_b L = 1 \times \nabla_Y L$$

2.1.4 激活层

激活函数（Activation functions）对于人工神经网络模型去学习、理解非常复杂和非线性的函数来说具有十分重要的作用。它们将非线性特性引入到我们的网络中。如图 1，在神经元中，输入的 inputs 通过加权，求和后，还被作用了一个函数，这个函数就是激活函数。引入激活函数是为了增加神经网络模型的非线性。没有激活函数的每层都相当于矩阵相乘。就算你叠加了若干层之后，无非还是个矩阵相乘罢了。 比较常

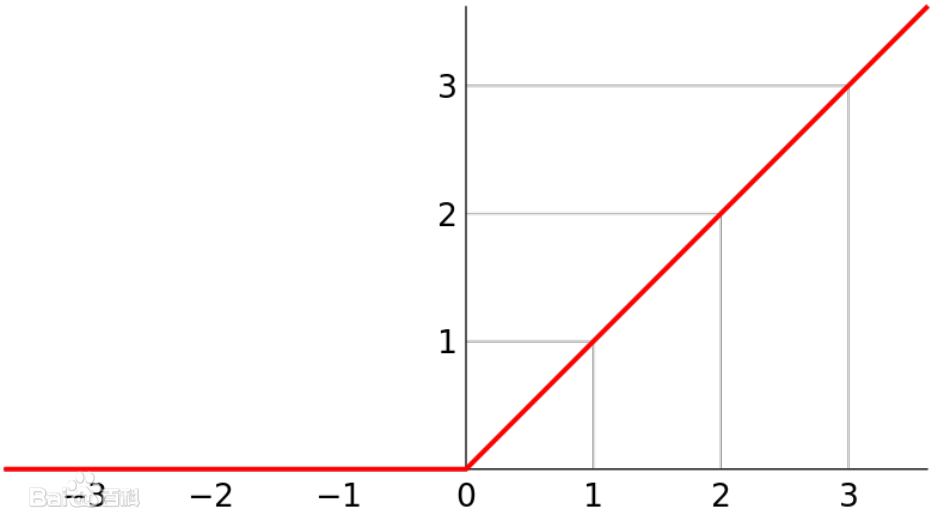

图 2-1 ReLU 激活函数

用的激活函数是 ReLU，其在 x 轴正半轴为正比例函数，负半轴为 0，公式可以表述为

$$ReLU(z) = max(0, z)$$

其反向传播过程可以表述为

2.1.5 L2 正则化

L2 正则化又叫做权值衰减，其操作方法是在损失项后面添加一个对权值的惩罚 $\frac{\lambda}{2}\|W\|$, 在反向传播过程中乘一个 $(1-\lambda)$ 即可。

3 代码细节

所有代码可以在 https://github.com/MengAaron/HW1-HandWrittingRecognition

3.0.1 网络构建

整体网络按照网络结构层、优化器、损失函数单独成文件，在构建网络的过程中，使用 yaml 文件进行网络参数的输入。防止了对代码文件的多次修改。网络结构在 yaml 文件中呈现格式为列表，列表中的每一项为字典，索引值为网络层名字、类型、以及参数。在 net.py 文件中循环将所有网络层进行整合，完成前向传播和后项传播的过程。

3.0.2 训练函数

引入 tqdm 进度条，训练函数输入网络、损失函数以及优化函数，一次就能完成整个更新过程。

3.0.3 参数类及其保存

对于每个参数，建立一个新的参数类，保存参数的值以及梯度。后续还可以引入 require_grad 参数，可以固定一些参数不参与更新。保存时候只需要保存参数值即可，使用 numpy 的 load 和 savez 函数。

4 实验结果与分析

4.1 数据集介绍

MNIST 是一个手写体数字的图片数据集，该数据集来由美国国家标准与技术研究所（National Institute of Standards and Technology (NIST)）发起整理，一共统计了来自 250 个不同的人手写数字图片。该数据集的收集目的是希望通过算法，实现对手写数字的识别。

4.2 实验结果分析

网络配置，我们采用层神经网络，隐藏层大小为 400，正则化强度为 0.001，学习率为 0.0001 每 5000 个循环下降一半，batch size 设为 16，分别绘制训练的 Loss 曲线和测试的 acc 曲线。 可以看到，训练的 Loss 在过了初始阶段下降后就一直在波动，而测试

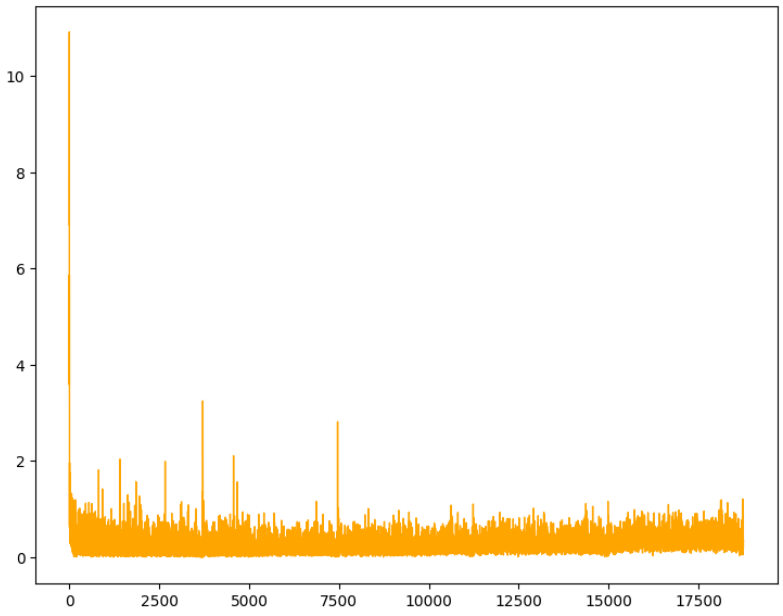

图 4-1 训练 loss 曲线

集的准确率在第二个 epoch 达到最高后就一直在下降，说明出现了严重的过拟合问题。

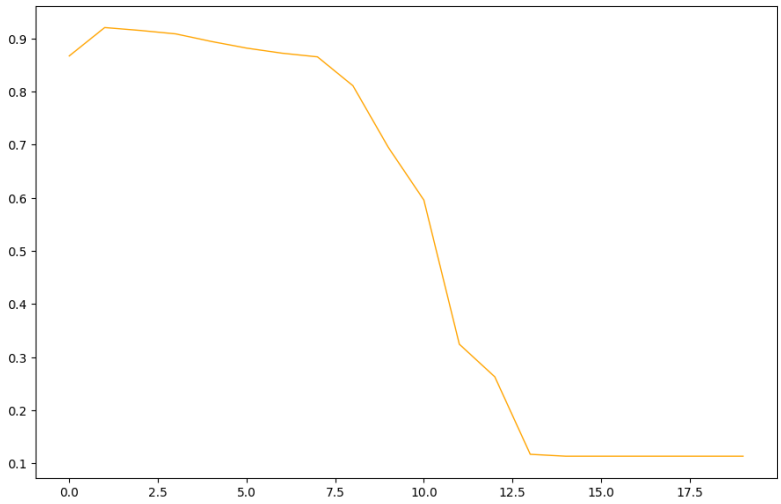

图 4-2 预测 acc 曲线

4.3 消融实验

4.3.1 学习率

固定网络结构和上面一样，学习率分别设置为 0.01，0.001,0.0001. 可以看到较小的学习率的结果越好。

lr	acc
0.01	0.1
0.001	0.88
0.0001	0.922

表 4-1 不同学习率对结果影响

4.3.2 隐藏层大小

保持学习率为 0.0001，隐藏层个数修改为 100,400,800，可以看到隐藏层适中结果比较好。

hidden	acc
100	0.915
400	0.922
800	0.919

表 4-2 不同学习率对结果影响

4.3.3 正则化强度

正则化强度个数修改为 0.1,0.01,0.001，可以看到较小的正则化强度得到的结果越好。

decay	acc
0.01	0.759
0.001	0.922
0.0001	0.968
0	0.971

表 4-3 不同正则化强度对结果影响