|  |  |
| --- | --- |
| **机器学习实验报告** | |
| 第 1 次 | |
| 图片包含 标牌  已生成极高可信度的说明 | |
| **姓名** | 何铭韬 |
| **班级** | 工设2204 |
| **学号** | 2223312318 |
| **电话** | 13034642319 |
| **Email** | 2780530464@qq.com |
| **日期** | 6.14 |

目录

[1实验题目 3](#_Toc198305898)

[2实验目的 3](#_Toc198305899)

[3实验环境与工具 3](#_Toc198305900)

[4实验背景与理论 3](#_Toc198305901)

[4.1数据集简介 3](#_Toc198305902)

[4.2神经网络和反向传播 3](#_Toc198305903)

[4.3激活函数 3](#_Toc198305904)

[4.4损失函数 3](#_Toc198305905)

[4.5优化算法 3](#_Toc198305906)

[5核心代码 4](#_Toc198305907)

[5.1 TODO 1 4](#_Toc198305908)

[5.2 TODO 2 4](#_Toc198305909)

[5.3 TODO 3 4](#_Toc198305910)

[5.4 TODO 4 4](#_Toc198305911)

[5.5 TODO 5 4](#_Toc198305912)

[5.6 TODO 6 4](#_Toc198305913)

[5.7 TODO 7 4](#_Toc198305914)

[6实验结果 4](#_Toc198305915)

[6.1损失与准确率曲线 4](#_Toc198305916)

[6.2超参数影响分析 4](#_Toc198305917)

[7其他方案 5](#_Toc198305918)

[8总结与收获 5](#_Toc198305919)

[附 录 5](#_Toc198305920)

# 1实验题目

使用全连接神经网络（两隐层）对 MNIST 手写数字进行分类识别

# 2实验目的

· 掌握从头实现神经网络的前向传播与反向传播算法。

· 理解交叉熵损失和 Softmax 输出的多分类模型原理。

· 熟悉利用小批量随机梯度下降（Mini-batch SGD）及超参数调优提升模型性能。

· 了解神经网络的完整结构，所需的层级，加权求和，激活函数。

# 3实验环境与工具

# 硬件环境：Intel® Core™ i5-13700kf，NVIDIA GeForce RTX 4070super, 32 GB 内存

# 操作系统：Windows 11 64 位

# 开发语言：Python 3.9

# 主要第三方库：

# numpy (矩阵计算)

# matplotlib (结果可视化)

# IDE：PyCharm 2024.14实验背景与理论

## 4.1数据集简介

MNIST 数据集由 NIST 提供的手写数字图像子集构成，共70000张 28×28 灰度图像，其中60000张用于训练，10000张用于测试。本实验将训练集再划分为 50000张训练、10000张验证数据

## 4.2神经网络和反向传播

**前向传播的线性变换：**本实验网络结构为784–192–30–10的全连接前馈网络，每层参数包括权重矩阵与偏置向量，如下为前向传播计算公式：

**反向传播:**

**输出层误差（BP1）:**使用交叉熵损失与 Softmax 输出时，输出层的误差极为简洁

**隐层误差递推（BP2）:**对于第 l 层（从倒数第二层到第一隐层），误差按照链式法则由上一层l+1传播回来

**梯度计算（BP3 & BP4）:**计算第l层的偏置和权重梯度，分别为

## 小批量累积与参数更新：在一个 mini-batch 内，将每个样本的累加

最后一次性按照带L2正则化的SGD规则更新参数（省略正则化项）

## 4.3激活函数

**隐藏层激活：**对线性变换结果应用 Sigmoid 激活函数，产生下一层的激活值，适用于隐藏层，导数为，最后一层采用Softmax将转换为概率分布，如下为激活函数公式：

其中：

Sigmoid：

Softmax：

## 4.4损失函数

**交叉熵损失（Cross‐Entropy）：**用于多分类时衡量预测概率与真实标签的差异。

能有效避免梯度消失，与 Softmax输出结合效果良好。

**二次代价（Quadratic Cost）：**也称均方误差，用于回归或对比实验

## 4.5优化算法

## 小批量随机梯度下降（Mini-batch SGD）＋L2 正则化：每次随机抽取若干样本计算梯度并更新参数；可选 L2 正则化控制过拟合。

# 5核心代码

## 5.1 TODO 1

**多分类交叉熵损失函数**

**class CrossEntropyCost(object):**

**@staticmethod**

**def fn(a, y):**

**# 返回 a 和标签 y 之间的多分类交叉熵损失**

**loss = -np.sum(y \* np.log(a + 1e-10))**

**return loss**

## 5.2 TODO 2

**前向传播（隐藏层部分）**

**def feedforward(self, a):**

**for b, w in zip(self.biases[:-1], self.weights[:-1]):**

**a = sigmoid(np.dot(w, a) + b)**

**b, w = self.biases[-1], self.weights[-1]**

**a = softmax(np.dot(w, a) + b)**

**return a**

## 5.3 TODO 3

**反向传播中前向传播记录**

**activation = x**

**activations = [x]**

**zs = []**

**for b, w in zip(self.biases, self.weights):**

**z = np.dot(w, activation) + b**

**zs.append(z)**

**activation = sigmoid(z)**

**activations.append(activation)# 最后用 softmax 替换输出层激活**

**activations[-1] = softmax(zs[-1])**

## 5.4 TODO 4

**BP1：输出层误差计算**

**delta = self.cost.delta(zs[-1], softmax(zs[-1]), y)**

**nabla\_b[-1] = delta**

**nabla\_w[-1] = np.dot(delta, activations[-2].transpose())**

## 5.5 TODO 5

**BP2：隐藏层误差递推**

**for l in range(2, self.num\_layers):**

**z = zs[-l]**

**sp = sigmoid\_prime(z)**

**delta = np.dot(self.weights[-l+1].transpose(), delta) \* sp**

## 5.6 TODO 6

**BP3 & BP4：隐藏层梯度**

**nabla\_b[-l] = delta**

**nabla\_w[-l] = np.dot(delta, activations[-l - 1].transpose())**

## 5.7 TODO 7

**累积小批量梯度**

**nabla\_b = [nb + dnb for nb, dnb in zip(nabla\_b, delta\_nabla\_b)]**

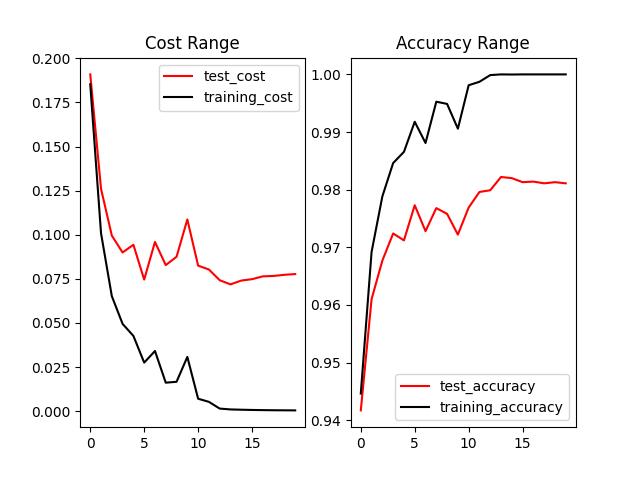
**nabla\_w = [nw + dnw for nw, dnw in zip(nabla\_w, delta\_nabla\_w)]**

# 6实验结果

## 6.1损失与准确率曲线

**默认参数：epochs=20，mini\_batch\_size=10，eta=0.5，隐藏层神经元：[192,30]，lmbda=0.0**

**结果如图所示：**

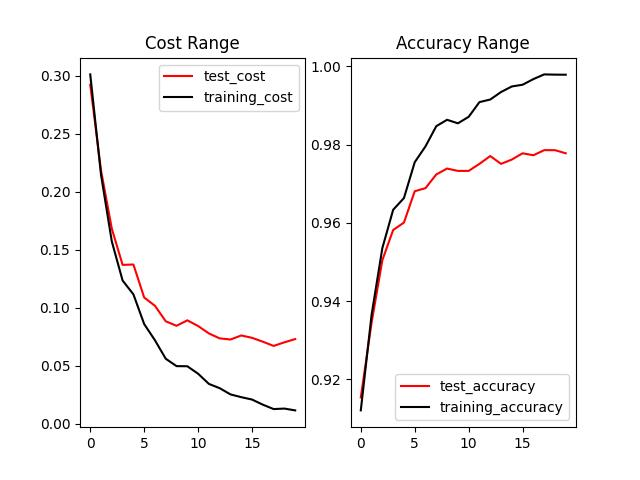
****

**结果描述：**训练集损失在前 5 个epoch迅速下降至0.05以下，验证集准确率稳定提升至约97.1%；后期损失收敛，准确率略有波动，最终准确率提升至约98.1%。

## 6.2超参数影响分析

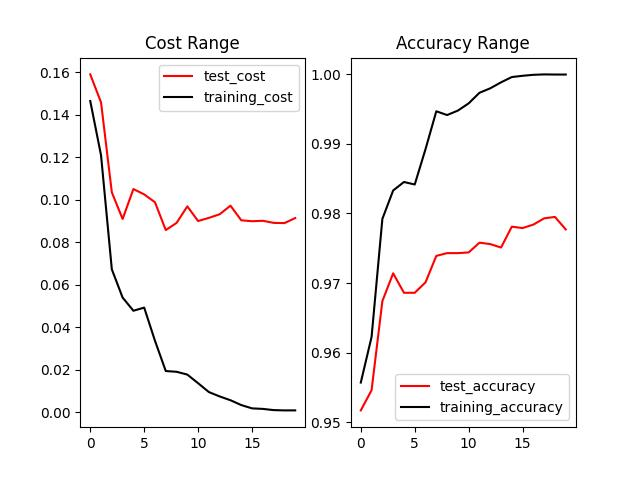
**组合1：eta=0.1**

较低学习率导致收敛更平缓，训练损失在 20 个 epoch 后仍未达到默认参数的最低水平；准确率约为 97.8%。



**组合2：hidden\_dim=[100,20]**

隐层规模缩小后模型表达能力下降，验证准确率最好在 20 个 epoch 时为97.7%，收敛速度略有加快。



**结论：**

**关于学习率（eta）：**较高学习率（0.5）有助于快速收敛，但过高可能导致训练不稳定；

**关于神经元：**隐层神经元数量与模型容量成正比，应根据数据集规模权衡。过少（例如<30）会导致欠拟合，无法充分表达数据特征；过多（例如>200）则易过拟合、训练时间增加，且内存开销更大。

**关于训练轮数（epochs）**：训练轮数越多，模型有更多机会遍历整个训练集，损失和准确率通常能继续改善；但超过一定轮次后容易出现过拟合，且训练时间线性增长。

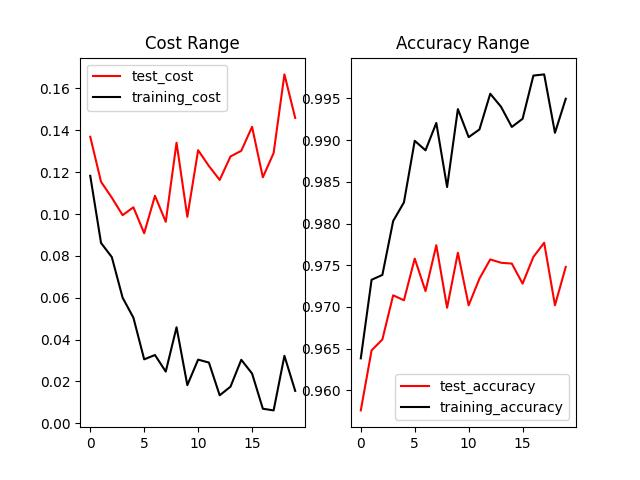
**关于小批量大小（mini\_batch\_size）**：较小的批量（如10–32）能更频繁地更新参数，加快对不同样本的探索，但梯度噪声大、收敛曲线抖动；较大的批量（如128–512）梯度估计平滑、GPU 利用率高，但也更易陷入局部最优且占用更多显存。

**关于隐层层数**：增加隐层层数可让模型分层提取更高级特征、提升表达能力；但层数过多易引发梯度消失或爆炸，需要 BatchNorm、残差连接等技巧来稳定训练；浅层网络则更易调试但表达能力有限。

# 7其他方案

（选择至少1种其他方案进行实验并分析，如选择不同的激活函数、权重初始化方法、损失函数、优化器、学习率衰减等。）

**激活函数替换为ReLU**：实验将隐藏层激活函数改为ReLU，再次训练可在相同超参数下略微提升收敛速度，并减少梯度消失问题。实验发现出现准确率和损失上下波动较大的情况



# 8总结与收获

手动实现反向传播加深了对神经网络训练原理的理解。

Cross‐Entropy 与 Softmax 结合可有效应对多分类任务。

超参数（学习率、隐层规模）对模型性能影响显著。

可尝试更多优化算法（如 Adam、学习率衰减）进一步提升效果。

# 附 录

如：

**6.1 数据结果展示**

**方案1参数（默认参数）：epochs=20，mini\_batch\_size=10，eta=0.5，隐藏层神经元：[192,30]，lmbda=0.0**

Epoch 17 training complete

Cost on training data: 0.0005069308595437162

Accuracy on training data: 50000.0 / 50000

Cost on evaluation data: 0.07662928619464676

Accuracy on evaluation data: 9811.0 / 10000

Epoch 18 training complete

Cost on training data: 0.00047110018108379776

Accuracy on training data: 50000.0 / 50000

Cost on evaluation data: 0.07729668728140617

Accuracy on evaluation data: 9813.0 / 10000

Epoch 19 training complete

Cost on training data: 0.0004255134693519969

Accuracy on training data: 50000.0 / 50000

Cost on evaluation data: 0.07769092628591452

Accuracy on evaluation data: 9811.0 / 10000

**方案2参数(改变学习率）：epochs=20，mini\_batch\_size=10，eta=0.1，隐藏层神经元：[192,30]，lmbda=0.0**

Epoch 17 training complete

Cost on training data: 0.01475198501478428

Accuracy on training data: 49865.0 / 50000

Cost on evaluation data: 0.07532197316430263

Accuracy on evaluation data: 9768.0 / 10000

Epoch 18 training complete

Cost on training data: 0.011585934085852845

Accuracy on training data: 49911.0 / 50000

Cost on evaluation data: 0.07164950809234502

Accuracy on evaluation data: 9783.0 / 10000

Epoch 19 training complete

Cost on training data: 0.010934085347896437

Accuracy on training data: 49932.0 / 50000

Cost on evaluation data: 0.07323807125225194

Accuracy on evaluation data: 9781.0 / 10000

**方案3参数（改变隐藏层）：epochs=20，mini\_batch\_size=10，eta=0.5，隐藏层神经元：[100,20]，lmbda=0.0**

Epoch 17 training complete

Cost on training data: 0.0009457564880392375

Accuracy on training data: 49999.0 / 50000

Cost on evaluation data: 0.0890840884877201

Accuracy on evaluation data: 9793.0 / 10000

Epoch 18 training complete

Cost on training data: 0.0008042863773384179

Accuracy on training data: 49998.0 / 50000

Cost on evaluation data: 0.08898387293689017

Accuracy on evaluation data: 9795.0 / 10000

Epoch 19 training complete

Cost on training data: 0.0008144533354050667

Accuracy on training data: 49998.0 / 50000

Cost on evaluation data: 0.09134915662998458

Accuracy on evaluation data: 9777.0 / **10000**

**方案4参数（隐藏层激活函数改为relu）：epochs=20，mini\_batch\_size=10，eta=0.5，隐藏层神经元：[192,30]，lmbda=0.0**

Epoch 17 training complete

Cost on training data: 0.006151295138882492

Accuracy on training data: 49894.0 / 50000

Cost on evaluation data: 0.1291637218322086

Accuracy on evaluation data: 9777.0 / 10000

Epoch 18 training complete

Cost on training data: 0.03224692566615349

Accuracy on training data: 49544.0 / 50000

Cost on evaluation data: 0.16673268874546684

Accuracy on evaluation data: 9702.0 / 10000

Epoch 19 training complete

Cost on training data: 0.015473916864847271

Accuracy on training data: 49748.0 / 50000

Cost on evaluation data: 0.14593269016640473

Accuracy on evaluation data: 9748.0 / 10000