# 机器学习实验报告

实验名称：建立全连接神经网络

学生：谢兴

学号：58122304

日期：2024/4/22

指导老师：刘胥影

助教：田秋雨

# 目录



# 1 任务描述

通过两种方式实现全连接神经网络，并对图片分类任务行进行测试与实验。

1. 手动实现简单的全连接神经网络

2. 使用 Pytorch 库简洁实现全连接神经网络

Fashion-MNIST 图片分类数据集包含 10 个类别的时装图像，训练集有 60,000 张图片，测试集中有 10,000 张图片。图片为灰度图片，高度（h）和宽度（w）均为 28 像素，通道数（channel）为 1。10 个类别分别为：t-shirt(T 恤), trouser（裤子）, pullover（套衫）, dress（连衣裙）, coat（外套），sandal（凉鞋），shirt（衬衫），sneaker（运动鞋），bag（包），ankle boot（短靴）。使用训练集数据进行训练，测试集数据进行测试。

# 2 教学要求

1. 掌握多层前馈神经网络及 BP 算法的原理与构建

2. 了解 PyTorch 库，掌握本实验涉及的相关部分

3. 进行参数分析实验，理解学习率等参数的影响

# 3 实验要求

## 3.1 使用 Python 编程构建手动实现单隐层全连接神经网络

**模型架构**

- 输入层 $28 \times 28 = 784$ 个节点，输出层 10 个节点，隐藏层 256 个节点。注意，可以将这两个变量都视为超参数。通常选择 2 的若干次幂作为层的宽度。因为内存在硬件中的分配和寻址方式，这么做往往可以在计算上更高效。
- 激活函数：ReLU 函数
- 损失函数：Cross entropy
- 性能指标：准确率
- 优化算法：实现标准 BP 或小批量梯度下降算法均可

**实现内容**

1. 初始化模型参数：对于每一层都要记录一个权重矩阵和一个偏置向量。
2. 设置激活函数：使用 ReLU 函数作为激活函数，要求手动实现该函数。
3. 前向计算：实现该函数。注意：需要将每个二维图像转化为向量进行操作。
4. 设置损失函数：使用 cross entropy 作为损失函数。可以自己手动实现，也可以直接调用 `nn.CrossEntropyLoss` 函数。
5. 训练模型：
   (a) 实现训练函数：该训练函数将会运行多个迭代周期（由 *num_epochs* 指定）。在每个迭代周期结束时，利用 *test_iter* 访问到的测试数据集对模型进行评估。利用后面给出的 Animator 类来可视化训练进度。
   (b) 可以使用 PyTorch 内置的优化器（`torch.optim.SGD`），也可以使用自己定制的优化器。
   (c) 可以调用 `torch.optim.SGD` 函数进行参数更新。

(d) 迭代周期数 *epoch* 设置为 10，学习率设置为 0.1，训练模型。

6. 设置性能函数：使用准确率 *accuracy* 作为性能指标。实现该函数。

7. 模型评估：

   (a) 对测试集数据进行测试。

   (b) 进行性能评估。

8. 参数分析实验：

   (a) 在所有其他参数保持不变的情况下，更改超参数 *num_hiddens* 的值，并查看此超参数的变化对结果有何影响。确定此超参数的最佳值。

   (b) 改变学习速率会如何影响结果？保持模型架构和其他超参数（包括轮数）不变，学习率设置为多少会带来最好的结果？

## 3.2 使用 PyTorch 库简洁实现全连接神经网络

手动实现一个简单的多层神经网络是很容易的。然而如果网络有很多层，从零开始实现会变得很麻烦。可以使用高级 API 如 PyTorch 库简洁实现。

1. 请使用 PyTorch 库简洁实现前述的全连接神经网络，并进行模型评估。

   (a) 优化器：使用 `torch.optim.SGD`

   (b) 小批量数据载入函数参见提供的代码。

2. **参数分析实验**

   (a) 尝试添加不同数量的隐藏层（也可以修改学习率），怎么样设置效果最好？

   (b) 尝试不同的激活函数，哪个效果最好？

   (c) 尝试不同的方案来初始化权重，什么方法效果最好？

## 3.3 提交要求

其中报告内容包括以下几个部分：

1. 手动实现单隐层全连接神经网络

   (a) 训练过程中，训练集与验证集误差随 epoch 变化的曲线图

(b) 性能评估结果

(c) 参数分析实验：包括实验设置与结果分析

2. 使用 PyTorch 库简洁实现全连接神经网络

(a) 性能评估结果

(b) 参数分析实验：包括实验设置与结果分析

# 4 数据集

# 5 训练与测试

# 6 实验总结

# 附录 A

## 附录 A1

```python
import pandas as pd
import numpy as np
import seaborn as sns
import pickle
from sklearn.preprocessing import MinMaxScaler
from statsmodels.stats.outliers_influence import
variance_inflation_factor
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from statsmodels.stats.outliers_influence import
variance_inflation_factor
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
# 读取 CSV 文件
df = pd.read_csv('song_data.csv')
# # 标记重复行（完全相同的行视为重复）
# df['is_duplicate'] = df.duplicated(keep=False) # keep=False 会标记所
有的重复项
#
# # 计算重复与非重复的行数
# duplicate_distribution = df['is_duplicate'].value_counts()
#
# # 绘制重复行分布的条形图
# plt.figure(figsize=(8, 6))
# duplicate_distribution.plot(kind='bar')
# plt.title('Distribution of Duplicate Rows')
# plt.xticks(ticks=[0, 1], labels=['Unique Rows', 'Duplicate Rows'],
rotation=0)
# plt.ylabel('Count')
# plt.xlabel('Row Type')
# plt.show()
```

Listing 1: Python example

# 附录 B

## 附录 B1