# 模式识别与机器学习大作业

姓名：黄梓赫
学号：202428013229070

## 一．课程要求

从四类方法中选三类方法，从选定的每类方法中，各选一种具体的方法，从给定的三个数据集中选一个数据集对这三种方法进行测试比较。

第一类方法：线性方法：线性 SVM、 Logistic Regression
第二类方法：非线性方法：Kernel SVM， 决策树
第三类方法：集成学习：Bagging， Boosting
第四类方法：神经元网络：自选结构

**实现部分：**

在 MNIST 数据集中分别使用 Logistic Regression，决策树以及 CNN 方法进行实验

## 二．实验方法

1．Logistic Regression 方法

方法介绍：逻辑回归通过建立输入特征与输出类别之间的关系来进行分类。它使用逻辑函数（sigmoid 函数）将线性组合的输入映射到 0 到 1 之间的概率值。

实现流程：导入 PyTorch 库，定义 logisticRg 类，创建线性层（输入 784 维，输出 10 维）。在 forward 方法中实现前向传播。根据选择的 method 初始化模型并移动到 GPU，同时定义 Adam 优化器和交叉熵损失函数。在训练过程中，遍历训练数据，展平输入，进行前向传播、计算损失、反向传播和参数更新。使用测试数据进行预测，计算准确率并输出结果。最后，输出训练和测试的最终结果。通过这些步骤，可以有效实现逻辑回归模型。

```python
import os
import torch
import torch.nn as nn
import torch.utils.data as Data
import torchvision

# Logistic Regression方法

class logisticRg(nn.Module):
    def __init__(self):
        super(logisticRg, self).__init__()
        self.lr = nn.Sequential(
            nn.Linear(28*28,10)
        )

    def forward(self, x):
        output = self.lr(x)
        return output, x
```

实验结果：

| epoch | batch_size | lr | ACC1 | ACC2 | ACC3 | ACC_aver |
|---|---|---|---|---|---|---|
| 1 | 32 | 1e-2 | 91.16% | 91.25% | 91.20% | 91.20% |
| 1 | 32 | 1e-3 | 91.75% | 91.46% | 91.62% | 91.61% |
| 1 | 32 | 1e-4 | 86.68% | 86.57% | 86.65% | 86.63% |
| 1 | 16 | 1e-3 | 91.69% | 91.65% | 91.94% | 91.76% |
| 1 | 64 | 1e-3 | 90.92% | 90.91% | 90.69% | 90.84% |
| 1 | 128 | 1e-3 | 90.11% | 89.46% | 0.8986 | 89.81% |
| ... | ... | ... | ... | ... | ... | ... |
| 5 | 32 | 1e-2 | 91.20% | 91.60% | 91.36% | 91.39% |
| 5 | 32 | 1e-3 | 92.72% | 92.62% | 92.77% | 92.70% |
| 5 | 32 | 1e-4 | 91.09% | 91.13% | 91.10% | 91.11% |
| 5 | 16 | 1e-3 | 92.57% | 92.51% | 92.61% | 92.56% |
| 5 | 64 | 1e-3 | 92.40% | 92.54% | 92.39% | 92.44% |
| 5 | 128 | 1e-3 | 92.25% | 92.28% | 92.09% | 92.21% |
| ... | ... | ... | ... | ... | ... | ... |

结果分析：实验结果表明，学习率和批量大小是影响模型性能的重要超参数。在第 1 个 epoch 中，学习率为 1e-3 时模型表现最佳，但当学习率降低到 1e-4 时，准确率显著下降。第 5 个 epoch 中，学习率为 1e-3 和批量大小为 32 的组合使得准确率超过 92%，显示出模型的稳定性和收敛性。整体来看，随着训练的进行，模型准确率逐渐提高，表明模型在不断优化。

2．决策树方法

方法介绍：决策树是一种常用的监督学习算法，适用于分类和回归任务。它通过将数据集分割成更小的子集，形成树状结构来进行决策。每个内部节点表示一个特征的测试，每个分支代表测试结果，而每个叶子节点则表示最终的类别或预测值。决策树的优点包括易于理解和解释、处理缺失值的能力以及不需要特征缩放等。

实现流程：首先导入必要的库，如 DecisionTreeClassifier 和 accuracy_score。在主训练代码中，通过判断 method 是否为'tree'来选择决策树模型，并初始化 DecisionTreeClassifier。接着，使用训练数据 X_train 和 y_train 调用 fit 方法训练模型。随后，对测试数据 test_x 进行预测，展平为适合模型输入的格式，并使用 predict 方法获取预测结果。最后，通过 accuracy_score 计算模型在测试集上的准确率，并打印结果以评估模型性能。

```
elif method == 'tree':
    model_method.fit(X_train, y_train)  # 训练决策树
    pred_y = model_method.predict(test_x.view(-1,28*28))  # 预测测试数据
    accuracy = accuracy_score(test_y, pred_y)  # 计算准确率
    print('Decision Tree | test accuracy: %.4f' % accuracy)
```

实验结果：

| Decision Tree | ACC1 | ACC2 | ACC3 | ACC4 | ACC5 | ACC_avg |
|---|---|---|---|---|---|---|
| | 87.91% | 87.74% | 88.03% | 88.10% | 87.90% | 87.73% |

结果分析：在本次实验中，决策树模型的表现通过多个准确率指标评估，最终得出平均准确率为 87.73%。各个准确率指标均在 87%至 88%之间，显示出模型在不同测试集上的稳定性和良好的分类性能。

3．CNN 方法

方法介绍：卷积神经网络（CNN）是一种深度学习模型，广泛应用于图像处理和计算机视觉任务。CNN 通过卷积层、激活层和池化层的组合，能够自动提取图像特征，减少手动特征工程的需求。其主要优点包括对局部特征的敏感性、参数共享和空间不变性，使得 CNN 在图像分类、目标检测等任务中表现优异。

实现流程：首先导入必要的 PyTorch 库，并创建一个名为 CNN 的类，继承自 nn.Module。在__init__方法中，构建两个卷积层，第一层接收 1 个输入通道，输出 16 个通道，卷积核大小为 5，步幅为 1，填充为 2，后接 ReLU 激活函数和 2x2 的最大池化层；第二层接收 16 个输入通道，输出 32 个通道，结构相似。接着，定义一个全连接层，将卷积层的输出展平后连接到 10 个输出节点。最后，在 forward 方法中，依次通过卷积层进行前向传播，展平输出并通过全连接层得到最终结果。通过这些步骤，CNN 模型能够有效地进行图像分类任务，自动提取特征并进行分类。

```python
# CNN 模块实现

class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Sequential(         # input shape (1, 28, 28)
            nn.Conv2d(
                in_channels=1,              # input height
                out_channels=16,            # n_filters
                kernel_size=5,              # filter size
                stride=1,                   # filter movement/step
                padding=2,                  # if want same width and length of this image after Conv2d, padding=(kernel_size-1)/2 if stride=1
            ),                              # output shape (16, 28, 28)
            nn.ReLU(),                      # activation
            nn.MaxPool2d(kernel_size=2),    # choose max value in 2x2 area, output shape (16, 14, 14)
        )
        self.conv2 = nn.Sequential(         # input shape (16, 14, 14)
            nn.Conv2d(16, 32, 5, 1, 2),     # output shape (32, 14, 14)
            nn.ReLU(),                      # activation
            nn.MaxPool2d(2),                # output shape (32, 7, 7)
        )
        self.out = nn.Linear(32 * 7 * 7, 10)   # fully connected layer, output 10 classes

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = x.view(x.size(0), -1)           # flatten the output of conv2 to (batch_size, 32 * 7 * 7)
        output = self.out(x)
        return output, x    # return x for visualization
```

实验结果：

| epoch | batch_size | lr | ACC1 | ACC2 | ACC3 | ACC_avg |
|---|---|---|---|---|---|---|
| 1 | 32 | 1e-2 | 98.01% | 97.35 | 97.59% | 97.65% |
| 1 | 32 | 1e-3 | 98.22% | 98.14% | 98.50% | 98.28% |
| 1 | 32 | 1e-4 | 94.41% | 94.26% | 94.13% | 94.26% |
| 1 | 16 | 1e-3 | 98.59% | 98.43% | 98.75% | 98.59% |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 64 | 1e-3 | 98.19% | 98.08% | 97.96% | 98.07% |
| 1 | 128 | 1e-3 | 97.86% | 97.22% | 97.50% | 97.52% |
| ... | ... | ... | ... | ... | ... | ... |
| 5 | 32 | 1e-2 | 98.08% | 97.96% | 97.76% | 97.93% |
| 5 | 32 | 1e-3 | 99.06% | 98.97% | 99.11% | 99.04% |
| 5 | 32 | 1e-4 | 98.16% | 98.15% | 98.12% | 98.14% |
| 5 | 16 | 1e-3 | 99.02% | 99.12% | 99.26% | 99.13% |
| 5 | 64 | 1e-3 | 99.07% | 98.88% | 98.95% | 98.96% |
| 5 | 128 | 1e-3 | 99.02% | 98.91% | 98.79% | 98.90% |
| ... | ... | ... | ... | ... | ... | ... |

结果分析：卷积神经网络（CNN）在 MNIST 数据集上表现优异，主要得益于其局部连接和权重共享特性，有效捕捉图像局部特征并减少参数数量；多层结构使得模型逐层提取特征；非线性激活函数（如 ReLU）增强了模型的表达能力；池化层降低特征图维度，提高计算效率并增强平移不变性。此外，MNIST 数据集的简单性和丰富的训练样本（60,000 个）为模型提供了充足的学习数据，现代优化算法加速了收敛过程。在实验中，学习率和批量大小显著影响模型性能，较高的学习率（如 1e-2）能快速收敛，而过低的学习率（如 1e-4）则导致准确率下降。批量大小为 32 时，学习率为 1e-3 的情况下，平均准确率达到 98.28%。随着 epoch 的增加，模型准确率逐渐提高，尤其在第 5 个 epoch 中超过 99%。尽管 CNN 表现优异，但仍有优化空间，如数据增强和正则化技术。