模式识别与统计学习实验三：逻辑回归

张力群 2020080301001

一、 实验原理

1.1 问题描述

逻辑回归是一种常用的分类算法，适用于二分类问题。在本次实验中，我们将使用逻辑回归算法来识别 MNIST 数据集中的数字 0 和数字 1。

MNIST 数据集是一个手写数字识别数据集，包含了 60,000 张训练图像和 10,000 张测试图像。每张图像都是 28 像素 x 28 像素的灰度图像，像素值的范围是 0 到 255。我们将使用该数据集来训练和测试逻辑回归模型。

1.2 逻辑回归原理

逻辑回归是一种基于概率的分类算法，它的原理是将输入特征X与权重参数 θ的线性组合，通过一个$sigmoid$函数进行激活，得到输出y的概率值。具体来说，如果y的概率值大于0.5，则分类为正例（数字1），否则分类为负例（数字0）。

逻辑回归的原理比较简单，主要分为两个部分：模型的构建和参数的优化。在模型的构建过程中，我们需要定义逻辑回归的假设函数（$hypothesisfunction$）和代价函数（$costfunction$）：

- 假设函数：

$$h_\theta(x) = g(\theta^T x)$$

  其中 θ 是权重参数，x 是输入特征
- $sigmoid$ 函数:

$$g(z) = 1/(1+e^{-z})$$

- 代价函数：

$$J(\theta) = [1/m] * sum(-y * log(g(\theta^T x)) - (1-y) * log(1-g(\theta^T x)))$$

  其中 m 是样本数量，y 是实际输出。

逻辑回归的核心就是求解最优参数，使代价函数最小化。在本实验中，我们将使用梯度下降算法实现参数优化过程。具体来说，我们需要求出代价函数的梯度，并更新参数 θ。具体的梯度计算公式为：

$$grad(\theta) = [1/m] * X^T * (g(X_\theta) - y)$$

其中 X 是输入特征矩阵，y 是实际输出，g 是 $sigmoid$ 函数。

通过不断迭代，我们可以逐步调整权重参数 θ，直至代价函数最小化。最终得到的 θ 即为逻辑回归模型的最优参数，可以用于预测和分类。

二、 python 代码实现

2.1 logistic_regression.py

```
# 用逻辑回归实现 MNIST 二分类

from sklearn.datasets import fetch_openml
from sklearn.preprocessing import StandardScaler
import numpy as np


# 定义 sigmoid 函数，用于计算激活值
def sigmoid(x):
    return 1 / (1 + np.exp(-x))


# 定义损失函数，用于计算负对数似然
def neg_log_likelihood(theta, X, y):
    theta = theta.reshape(-1, 1)  # 转换为列向量
    y = y.reshape(-1, 1)  # 转换为列向量
    z = np.dot(X, theta)  # 计算激活值
    return np.mean(-y * z + np.log(1 + np.exp(z)))  # 计算损失


# 定义梯度函数，用于计算梯度
def grad(theta, X, y):
    theta = theta.reshape(-1, 1)  # 转换为列向量
    y = y.reshape(-1, 1)  # 转换为列向量
    z = np.dot(X, theta)  # 计算激活值
    return 1 / X.shape[0] * np.dot(X.T, sigmoid(z) - y)  # 计算梯度


# 加载训练数据
print('Loading training data...')
X_train, y_train = fetch_openml('mnist_784', version=1, return_X_y=True,
as_frame=False, parser='auto')
X_train = X_train[(y_train == '0') | (y_train == '1')]  # 只取标签为0和1的数据
y_train = y_train[(y_train == '0') | (y_train == '1')]  # 只取标签为0和1的数据
scaler = StandardScaler()  # 初始化特征缩放器
X_train = scaler.fit_transform(X_train.astype(np.float32))  # 特征缩放
X_train = np.insert(X_train, 0, 1, axis=1)  # 插入常数项
y_train = y_train.reshape(-1, 1)  # 转换为列向量
y_train = y_train.astype(np.float32)  # 转换为浮点数

# 训练逻辑回归模型
theta0 = np.zeros((X_train.shape[1], 1))  # 初始化参数
learning_rate = 0.1  # 学习率
num_iterations = 500  # 迭代次数

print('Training...')
for i in range(num_iterations):
    theta0 -= learning_rate * grad(theta0, X_train, y_train)  # 梯度下降
```

```
# 加载测试数据
print('Loading testing data...')
X_test, y_test = fetch_openml('mnist_784', version=1, return_X_y=True,
as_frame=False, parser='auto')
X_test = X_test[(y_test == '0') | (y_test == '1')]  # 只取标签为0和1的数据
y_test = y_test[(y_test == '0') | (y_test == '1')]  # 只取标签为0和1的数据
X_test = scaler.transform(X_test.astype(np.float32))  # 特征缩放
X_test = np.insert(X_test, 0, 1, axis=1)  # 插入常数项
y_test = y_test.reshape(-1, 1)  # 转换为列向量
y_test = y_test.astype(np.float32)  # 转换为浮点数

# 在训练集和测试集上进行预测
print('Predicting...')
y_train_pre = (sigmoid(np.dot(X_train, theta0)) >= 0.5).astype(int)  # 预测训练
集
y_test_pre = (sigmoid(np.dot(X_test, theta0)) >= 0.5).astype(int)  # 预测测试集

# 计算分类准确度
train_accuracy = np.mean(y_train_pre == y_train)
test_accuracy = np.mean(y_test_pre == y_test)

print(f'Train accuracy: {train_accuracy:.4f}')
print(f'Test accuracy: {test_accuracy:.4f}')
```

1. 加载数据集，只取标签为0和1的数据，然后进行特征缩放和常数项插入。
2. 初始化参数，学习率和迭代次数。
3. 使用梯度下降算法训练模型。
4. 加载测试数据，进行特征缩放和常数项插入。
5. 在训练集和测试集上进行预测。
6. 计算分类准确度。

三、实验结果

1.分类准确度

```
/opt/anaconda/envs/data-science/bin/python /home/carton/workspace/python/Statistical-learning/homework/homework3/logistic_regression1.py
Loading training data...
Training...
Loading testing data...
Predicting...
Train accuracy: 0.9994
Test accuracy: 0.9994
```

从上图可以看出，逻辑回归模型在训练集上的分类准确度为 0.9994，在测试集上的分类准确度为 0.9994，说明逻辑回归模型具有很好的泛化能力。

四、总结体会

在本次实验中，我们使用逻辑回归算法来实现 MNIST 数据集的二分类问题（数字 0 和 1）。经过实验验证，我们获得了较高的分类准确率，说明逻辑回归算法对于简单的分类问题具有较好的性能。

在实验过程中，我深刻认识到了逻辑回归算法的原理和实现过程。逻辑回归算法是一种基于概率的分类算法，通过假设函数和代价函数来评估模型的性能，并通过梯度下降算法优化参数，得到最优的分类决策参数。

在实验中，我们使用 Python 编程语言实现了逻辑回归算法，并使用 MNIST 数据集进行测试和评估。实验结果表明，逻辑回归算法在处理 MNIST 二分类问题方面表现出较好的分类性能。

总之，在本次实验中，我深入理解了逻辑回归算法的原理和实现过程，并通过实验验证了其在分类问题中的优秀性能。这次实验不仅增加了我对机器学习的理解，同时也增强了我对 Python 编程语言的应用能力。