## Fisher分类实验报告

### 一、实验目标

本实验的目标是使用Fisher线性判别分析（LDA）对手写数字进行分类。特别是，我们关注于二分类问题，将数字1作为一个分类，而其它数字（2-9）作为另一个分类。

### 二、实验步骤与方案

\*\*1. 数据准备\*\*

我们使用了MNIST手写数字数据集，该数据集包含了0-9的手写数字，每个数字有大约7000个样本，每个样本是一个28x28的图像，展平后为784维的向量。

\*\*2. 数据预处理\*\*

为了将这个多分类问题转化为二分类问题，我们将标签转换为：'1' 或 '非1'。具体地，所有标签为'1'的样本被标记为1，其它所有样本被标记为0。

\*\*3. 数据集划分\*\*

我们将数据集分为训练集和测试集，其中训练集用于训练模型，测试集用于评估模型的性能。

\*\*方案1：使用Sklearn的LinearDiscriminantAnalysis\*\*

我们首先使用了Sklearn库中的LinearDiscriminantAnalysis进行分类。这是一个现成的工具，可以直接对数据进行训练和预测。

```python

# Fisher线性判别分析-Fisher LDA

# 手写数字识别的Fisher十分类Python程序

import numpy as np

from sklearn import discriminant\_analysis

from sklearn import datasets

from sklearn.metrics import accuracy\_score

np.random.seed(1000)

digit = datasets.load\_digits()

# 10类，每类180左右，共1797个样本，64维

digit\_x = digit.data

digit\_y = digit.target

indices = np.random.permutation(len(digit\_x))

# 留200个样本作为测试集，剩下是训练集（大概是9：1）

digit\_x\_train = digit\_x[indices[:-200]]

digit\_y\_train = digit\_y[indices[:-200]]

digit\_x\_test = digit\_x[indices[-200:]]

digit\_y\_test = digit\_y[indices[-200:]]

# 定义Fisher分类器对象fisher\_clf

fisher\_clf = discriminant\_analysis.LinearDiscriminantAnalysis()

# 调用该对象的训练方法

fisher\_clf.fit(digit\_x\_train, digit\_y\_train)

# 调用该对象的测试方法

digit\_y\_pred = fisher\_clf.predict(digit\_x\_test)

print('测试数据集的正确标签为:', digit\_y\_test)

print('测试数据集的预测标签为:', digit\_y\_pred)

testing\_acc = accuracy\_score(digit\_y\_test, digit\_y\_pred) \* 100

print('Fisher线性分类器测试准确率: {:.2f}%'.format(testing\_acc))

```

\*\*方案2：使用numpy实现\*\*

我们还自己实现了Fisher的LDA算法。具体步骤如下：

- 计算每个类的均值。

- 计算每个类的散布。

- 计算总的类内散布矩阵。

- 计算Fisher判别准则。

- 将数据投影到Fisher方向上。

- 根据投影值和阈值进行分类。

```python

import numpy as np

from sklearn.datasets import fetch\_openml

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# 加载MNIST数据集

mnist = fetch\_openml('mnist\_784')

X, y = mnist.data, mnist.target

# 将标签转换为二分类：'1' 或 '非1'

y\_bin = np.where(y == '1', 1, 0)

# 将数据集分为训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_bin, test\_size=0.2, random\_state=42)

def fisher\_lda(X, y):

# 计算每个类的均值

m1 = np.mean(X[y == 1], axis=0)

m2 = np.mean(X[y == 0], axis=0)

# 计算每个类的散布

S1 = np.dot((X[y == 1] - m1).T, (X[y == 1] - m1))

S2 = np.dot((X[y == 0] - m2).T, (X[y == 0] - m2))

# 计算总的类内散布矩阵

Sw = S1 + S2

# 计算Fisher判别准则

w = np.linalg.pinv(Sw).dot(m1 - m2)

return w

# 训练Fisher的LDA

w = fisher\_lda(X\_train, y\_train)

# 将数据投影到Fisher方向上

X\_train\_proj = X\_train.dot(w)

X\_test\_proj = X\_test.dot(w)

# 设置分类阈值

threshold = np.mean(X\_train\_proj)

# 预测类标签

y\_train\_pred = np.where(X\_train\_proj > threshold, 1, 0)

y\_test\_pred = np.where(X\_test\_proj > threshold, 1, 0)

# 计算准确率

train\_accuracy = accuracy\_score(y\_train, y\_train\_pred)

test\_accuracy = accuracy\_score(y\_test, y\_test\_pred)

print(f"训练准确率: {train\_accuracy:.4f}")

print(f"测试准确率: {test\_accuracy:.4f}")

```

### 三、实验结果

\*\*方案1的结果：\*\*

使用Sklearn的LinearDiscriminantAnalysis，我们得到了以下的分类准确率：

- Fisher线性分类器测试准确率: 97.00%

这意味着，我们的模型在测试集上有97%的准确率。

\*\*方案2的结果：\*\*

使用自己实现的Fisher LDA算法，我们得到了以下的分类准确率：

- 训练准确率: 80.22%

- 测试准确率: 79.88%

这意味着，我们的模型在训练集上有80.22%的准确率，在测试集上有79.88%的准确率。

### 四、实验总结

本实验中，我们使用了两种方案对手写数字进行了二分类。方案1使用了Sklearn库，而方案2则是自己实现的。从实验结果来看，方案1的性能明显优于方案2。

这可能是因为Sklearn的LDA实现包含了一些优化和正则化技巧，而我们自己的实现则是基于Fisher LDA的基本原理。尽管如此，我们自己的实现仍然达到了近80%的准确率，这是一个相当不错的结果。

总的来说，Fisher LDA是一个强大的工具，特别是在处理高维数据时。在未来的工作中，我们可以考虑引入更多的优化和正则化技巧，以进一步提高模型的性能。