西安电子科技大学

模式识别大作业

题　　目：Fisher判别分析

姓　　名：X X X

学　　号：2300920XXX

专　　业：

摘要

本报告实现了 Fisher 线性判别算法（通过线性判别分析，LDA），并采用 k 折交叉验证方法，在 UCI 数据集的 Iris 和 Sonar 数据集上验证算法有效性。实验结果表明，该算法在 Iris 数据集上表现优异，平均准确率较高；在 Sonar 数据集上也能取得一定准确率，整体能较好完成分类任务，但在高维复杂的 Sonar 数据上仍有提升空间。

关键词： Fisher 线性判别；k 折交叉验证；分类

一．绪论

在模式识别领域，分类是核心任务之一。Fisher 线性判别作为一种经典的线性分类方法，旨在寻找一个最优的投影方向，使得不同类别数据在该方向上的投影尽可能分离，从而实现分类。对于多类分类问题，线性判别分析（LDA）是 Fisher 线性判别的推广，它通过最大化类间散度与类内散度的比值来确定投影矩阵。

实际应用中，为了更准确评估算法性能，需要科学的验证方法。k 折交叉验证将数据集划分为 k 个互斥子集，依次用 k-1 个子集训练模型，剩余 1 个子集测试模型，最终取 k 次测试结果的平均，能有效减少因数据划分带来的偏差，更全面反映算法在数据集上的表现。本次实验选取 Iris（3 类、4 维、150 个样本）和 Sonar（2 类、60 维、208 个样本）数据集，利用 k 折交叉验证验证 Fisher 线性判别算法的分类性能。

二．算法介绍

K折交叉验证（K-Fold Cross-Validation）是一种常用的模型评估技术，广泛应用于机器学习和统计学中。其主要目的是通过多次训练和测试，评估模型的性能，减少过拟合和欠拟合的风险。

在K折交叉验证中，数据集被随机分成K个等大小的子集（称为折）。每次迭代中，选择一个子集作为验证集，其余K-1个子集作为训练集。整个过程重复K次，每个子集都有一次作为验证集的机会。最终的模型性能是K次测试结果的平均值。

K折交叉验证的步骤为：

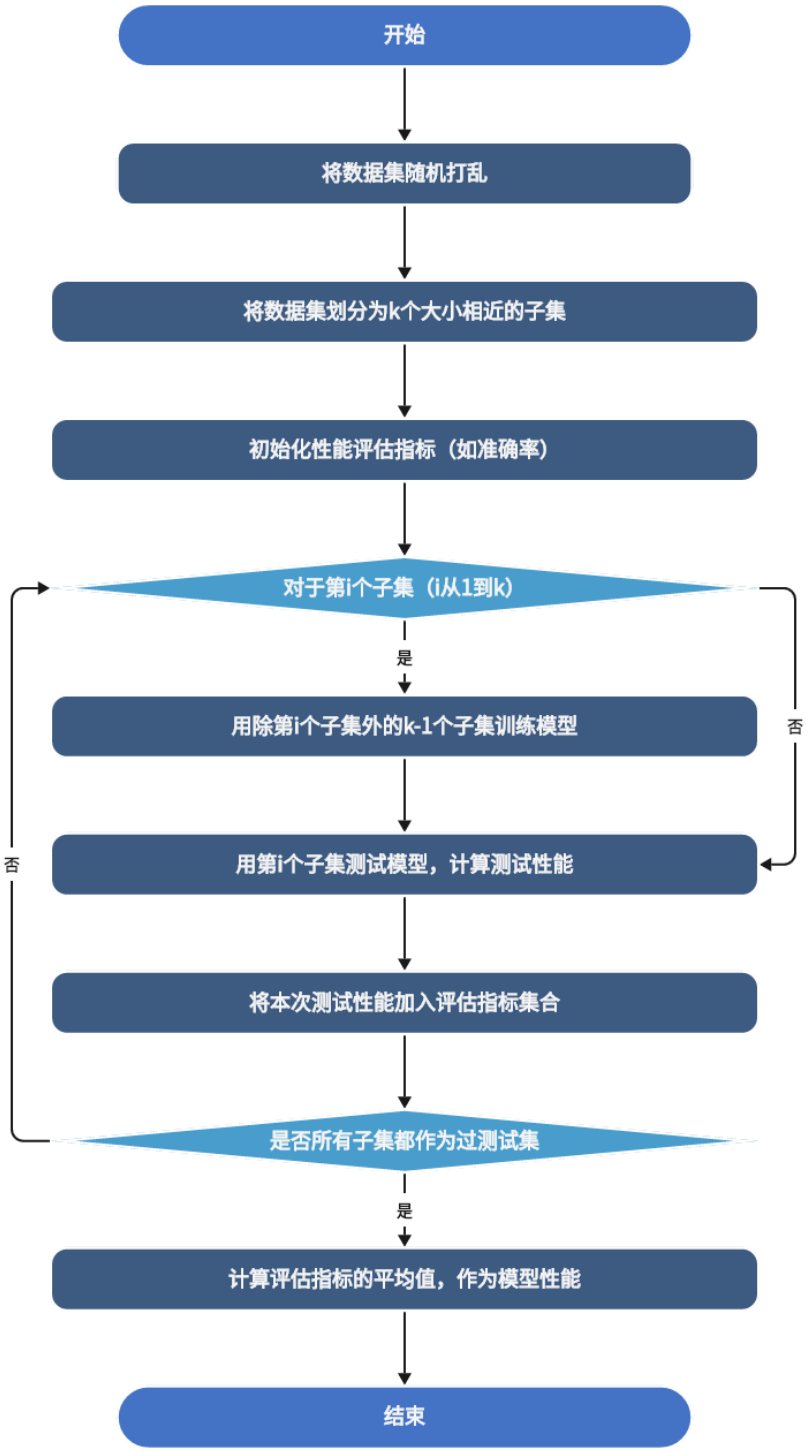
将数据集随机分成K个等大小的子集。

在每次迭代中，选择一个子集作为验证集，其余K-1个子集作为训练集。

使用训练集训练模型，并在验证集上评估模型性能。

重复上述过程K次，记录每次的评估结果。

计算K次评估结果的平均值，作为模型的最终性能指标。



三．实验过程

（一）数据加载与预处理：加载 Iris 和 Sonar 数据集，对特征数据进行标准化处理，使各特征具有相同的尺度，利于 LDA 算法工作。

（二）模型训练与验证：使用scikit-learn中的LinearDiscriminantAnalysis

实现 LDA 模型，采用 6 折交叉验证(k = 6)，分别对两个数据集进行验证，得到每次交叉验证的准确率。

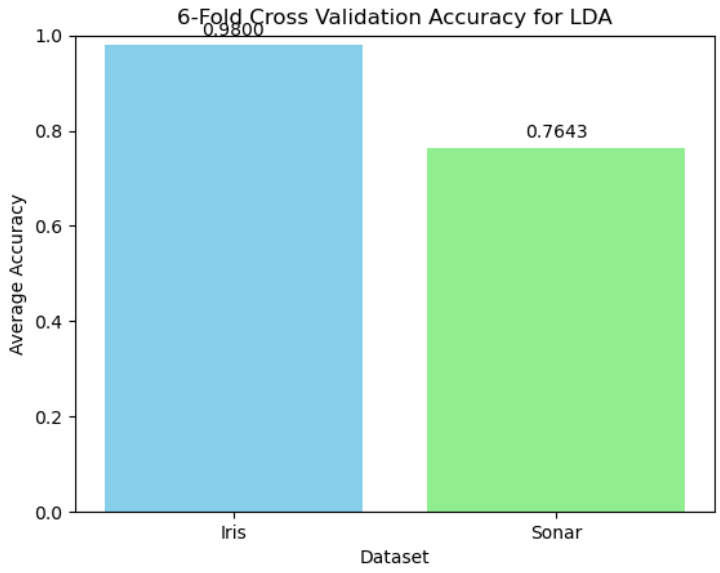
（三）结果计算与可视化：计算 6 次交叉验证准确率的平均值，绘制两个数据集上平均准确率的对比条形图。

（四）最终实验结果为：

```
D:\app\miniconda\envs\pytorch\python.exe D:\app\基本组件\Desktop\moshishibie\main.py
Iris 数据集 6-折交叉验证的平均准确率: 0.9800
Sonar 数据集 6-折交叉验证的平均准确率: 0.7643

进程已结束,退出代码0
```

（五）可视化结果为：



四．实验结论

（一）优势

Fisher 线性判别（LDA）算法作为经典的线性分类方法，原理清晰，在低维、类别区分度较好的数据集（如 Iris）上能取得很高的分类准确率，具有良好的线性分类能力。k 折交叉验证方法能有效减少数据划分带来的性能评估偏差，更全面、准确地反映算法在数据集上的泛化性能，为算法性能评估提供了可靠的手段。

（二）不足

LDA 作为线性分类算法，依赖数据的线性可分性假设，当数据存在非线性分布时（如 Sonar 数据集可能存在一定非线性特征），分类性能会受到限制。Sonar 数据集特征维度高，LDA 在高维空间中可能面临计算复杂度较高、以及 “维度灾难” 带来的一些问题，影响分类效果。

附录：代码

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score, KFold
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.utils import shuffle

# 加载 Iris 数据集
iris = load_iris()
X_iris, y_iris = iris.data, iris.target

# 加载 Sonar 数据集
# 数据格式为：前60列为特征，最后一列为标签（'R'表示岩石，'M'表示
水雷，转换为0和1）
sonar_data = np.genfromtxt('sonar.csv', delimiter=',',
dtype=None, encoding=None,skip_header=1)
X_sonar = np.array([list(row)[:-1] for row in sonar_data],
dtype=float)
y_sonar = np.array([0 if row[-1] == 'R' else 1 for row in
sonar_data])

# 数据预处理：标准化
scaler_iris = StandardScaler()
X_iris_scaled = scaler_iris.fit_transform(X_iris)

scaler_sonar = StandardScaler()
X_sonar_scaled = scaler_sonar.fit_transform(X_sonar)

# 定义 k 折交叉验证的折数
k = 6
```

```
# 对 Iris 数据集进行 k 折交叉验证
kf_iris = KFold(n_splits=k, shuffle=True, random_state=42)
lda_iris = LinearDiscriminantAnalysis()
iris_scores = cross_val_score(lda_iris, X_iris_scaled, y_iris,
cv=kf_iris)

# 对 Sonar 数据集进行 k 折交叉验证
kf_sonar = KFold(n_splits=k, shuffle=True, random_state=42)
lda_sonar = LinearDiscriminantAnalysis()
sonar_scores = cross_val_score(lda_sonar, X_sonar_scaled,
y_sonar, cv=kf_sonar)

# 计算平均准确率
iris_avg_accuracy = np.mean(iris_scores)
sonar_avg_accuracy = np.mean(sonar_scores)

# 输出结果
print(f"Iris 数据集 {k}-折交叉验证的平均准确率:
{iris_avg_accuracy:.4f}")
print(f"Sonar 数据集 {k}-折交叉验证的平均准确率:
{sonar_avg_accuracy:.4f}")

datasets = ['Iris', 'Sonar']
accuracies = [iris_avg_accuracy, sonar_avg_accuracy]

plt.bar(datasets, accuracies, color=['skyblue', 'lightgreen'])
plt.title(f'{k}-Fold Cross Validation Accuracy for LDA')
plt.xlabel('Dataset')
plt.ylabel('Average Accuracy')
plt.ylim(0, 1)
for i, acc in enumerate(accuracies):
    plt.text(i, acc + 0.02, f'{acc:.4f}', ha='center')
plt.show()
```