西安电子科技大学人工智能学院

模式识别 课程实践报告

成　绩

实践课题名称 ***Fisher*判别分析**

班级 2020039 姓名 刘焕宇 学号 20009200770

实践日期 2022 年 9 月

指导教师评语：
指导教师： 年 月 日

一、实验内容

线性判别式分析$(Linear\ Discriminant\ Analysis, LDA)$是模式识别的经典算法，在1996年由$Bellhumeur$引入人工智能和模式识别领域，是一种常用的数据降维基本方法，其基本思想是将高维的模式样本投影到最佳鉴别矢量空间，以达到抽取分类信息和压缩特征空间维数的效果，投影后保证模式样本在新的子空间有最大的类间距离和最小的类内距离，即模式在该空间中有最佳的可分离性。因此，它是一种有效的特征抽取方法。使用这种方法能够使投影后模式样本的类间离散度矩阵最大，并且同时类内离散度矩阵最小。就是说，它能够保证投影后模式样本在新的空间中有最小的类内距离和最大的类间距离，即模式在该空间中有最佳的可分离性。

费舍尔判别准则$(Fisher\ discrimination\ criterion, Fisher)$是$LDA$是在二分类问题上的特殊情况，选择一个适当的投影轴，使所有样本点都投影到这个轴上得到一个投影值，将多维问题转化为一维问题。在计算投影轴时，需保证每一类的投影距离尽可能小，而不同类间投影距离尽可能大，这与LDA是相同的。

本实验旨在学习应用LDA与$Fisher$，掌握其数据降维的思想与公式的推导。

二、实验原理

设 n 维样本 X 空间，第i类有N_i个样本，共有k类，映射集分别为$\varphi_1, \varphi_2, \dots \varphi_{cls}$，降维后$k-1$维类别 Y 空间：

1. 各类样本的均值向量

$$\mu_i = \frac{1}{N_i} \sum_{x_j \in \varphi_i} x_j , \qquad i = 1,2, \cdots, k$$

2. 样本类内离散度

$$S_i = \sum_{x_j \in \varphi_i} (x_j - \mu_i)\left(x_j - \mu_i\right)^T, \qquad i = 1,2, \cdots, k$$

3. 总样本类内离散度

$$S_w = \sum_{i=1}^{k} S_i$$

4. 总样本类间离散度

$$S_b = \sum_{i=1}^{k} (\mu_i - \mu)(\mu_i - \mu)^T$$

其中 μ 为所有样本均值.

以此类推，在投影后的Y空间也有相同的定义，我们用$\overline{S_w}, \overline{S_b}$来表示，具体公式与$n$维空间中的相同。

降维后，我们应该使样本间尽量分开，同时类内样本投影尽可能密集，由此构造准则函数为

$$J(w) = \frac{\overline{S_b}}{\overline{S_w}}$$

则目标为找到使$J(w)$最大的w^*，带入$y = w^T x$后，将其化简为包含w的显函数，可得到广义瑞利商的形式:

$$J\ (w) = \frac{w^T S_b w}{w^T S_w w}$$

构造拉格朗日函数求导可得:

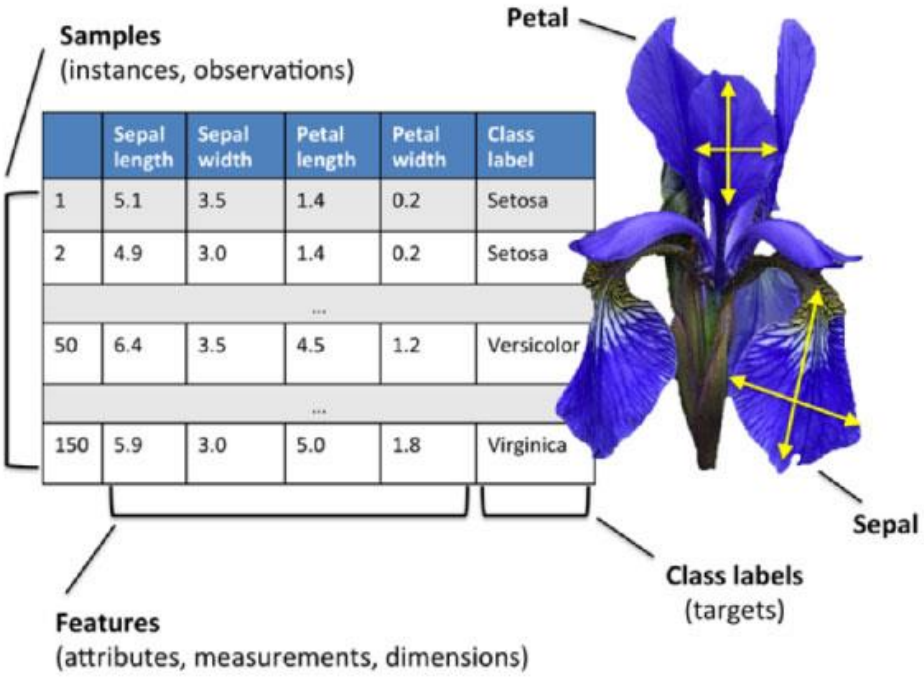
$$S_w^{-1} S_b w^* = \lambda w^*$$

我们取前$k-1$个最大的特征值对应的特征向量组成W。由于S_b是由k个均值矩阵μ_i的和，而μ_k可以由前 $k-1$个μ线性组合得到，故S_b的秩为$k-1$，降维的最大维度为$k-1$.

三、实验结果与分析

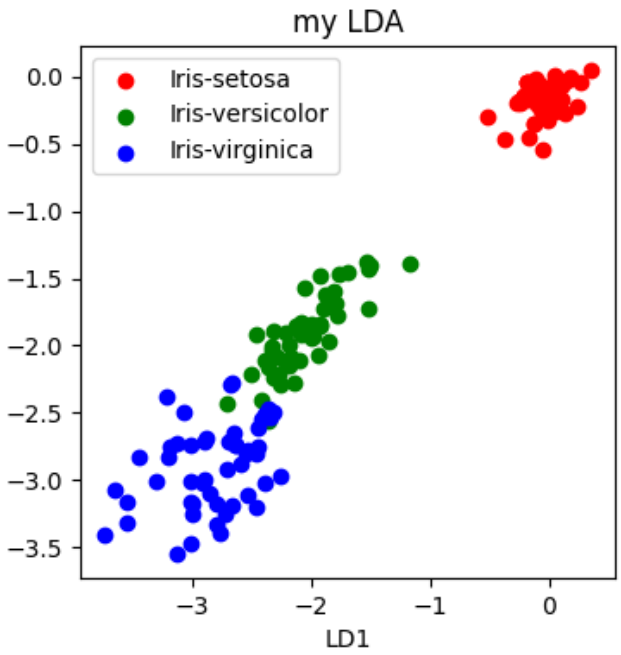
3.1 *LDA*在鸢尾花数据集的结果

Iris也称鸢尾花卉数据集，是一类多重变量分析的数据集。通过花萼长度，花萼宽度，花瓣长度，花瓣宽度4个属性预测鸢尾花卉属于（Setosa(山鸢尾)，Versicolour(杂色鸢尾)，Virginica(维吉尼亚鸢尾)）三个种类中的哪一类，每类50个样本，共计150个样本。



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

本实验采用将数据随机分训练和测试，以9:1的比例随机划分，多次平均求结果，其中降成二维后的图像如下：

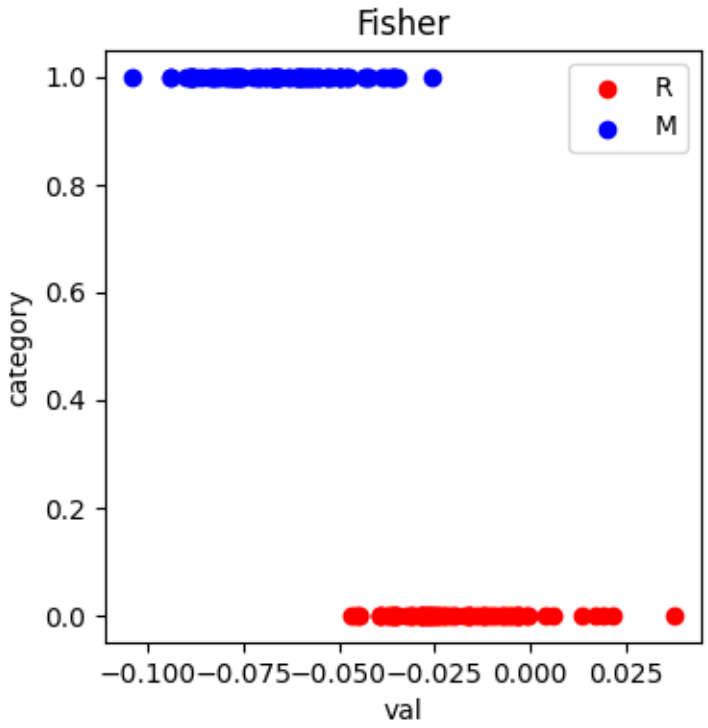


从图中可以看出，Iris数据集在两个方向上的投影分割比较明显，重叠部分较少，这印证了之后分类算法在Iris数据集上准确率较高。之后的分类任务，对于每一类计算出其在二维空间中的类别中心，计算各中心距待测样本点的欧氏距离，最近的中心即为其类别。多次划分数据集和测试集后，平均分类正确率为96.617%

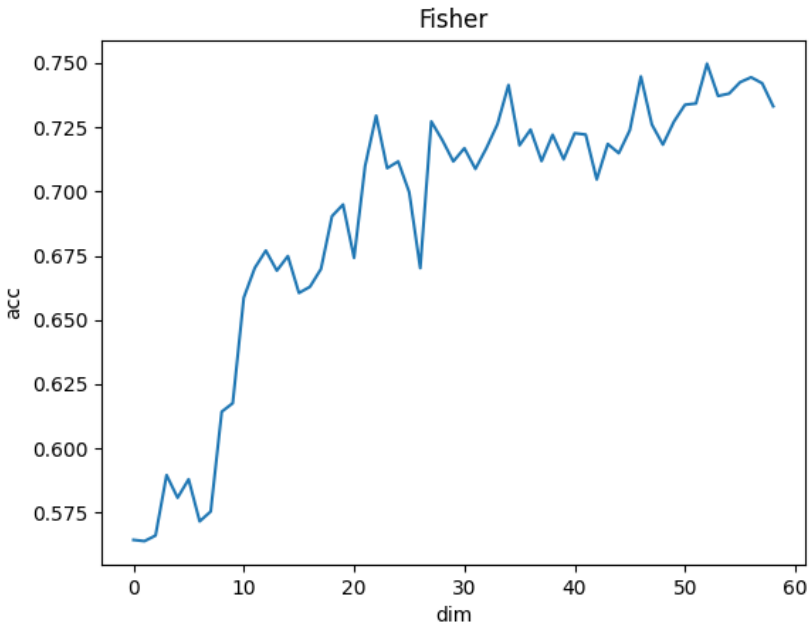
3.2 *Fisher*在*Sonar*数据集的结果

*Sonar*声呐数据集共有208行60列特征，数据分为两类，标签为R/M。表示208个观察对象，60个不同角度返回的力度值，二分类结果是岩石/金属。本实验聚焦于分析选择不同特征、不同维数对分类结果的影响。

首先取所有60维特征进行降维，做出其二维散点图，平均正确率为73.589%



之后根据不同数据维度，重复一百次使用随机划分训练集和测试集的方法计算平均准确率，作出维度与正确率的曲线图。



可以看出，随着维度的身高，准确率逐渐上升，但上升趋势越来越慢，且波动性较大，维度的升高会大幅提高计算的时间与复杂度，进而提高运行成本。

四、源程序代码

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random


def work(Iris, test):
    print(len(Iris[0]))
    mean = np.zeros((cls, dim))
    for i in range(cls):
        mean[i] = np.mean(Iris[i], axis=0)
        print('Mean Vector class %s:%s' % (i, mean[i]))

    # 类内离散度
    Si = np.zeros((cls, dim, dim))
    for i in range(cls):
        for x in Iris[i]:
            dx = (x - mean[i]).reshape((dim, 1))
            Si[i] += np.matmul(dx, dx.T)
    Sw = np.sum(Si, axis=0)
```

```
    print("类内离散度 Sw: \n", Sw)

    # 类间离散度
    mu = np.mean(mean, axis=0)  # 总平均值
    Sb = np.zeros((dim, dim))
    for i in range(cls):
        dx = (mean[i] - mu).reshape((dim, 1))
        Sb += 50 * np.matmul(dx, dx.T)
    print("类间离散度 Sw: \n", Sb)

    eigvals, eigvecs = np.linalg.eig(np.linalg.inv(Sw) * Sb)  # 求特征值，特征向量
    np.testing.assert_array_almost_equal(np.mat(np.linalg.inv(Sw) * Sb) * np.mat(eigvecs[:, 0].reshape(4, 1)),
                                         eigvals[0] * np.mat(eigvecs[:, 0].reshape(4, 1)), decimal=6, err_msg='',
                                         verbose=True)
    # sorting the eigenvectors by decreasing eigenvalues
    eig_pairs = [(np.abs(eigvals[i]), eigvecs[:, i]) for i in range(len(eigvals))]
    eig_pairs = sorted(eig_pairs, key=lambda k: k[0], reverse=True)
    W = np.hstack((eig_pairs[0][1].reshape(dim, 1), eig_pairs[1][1].reshape(dim, 1)))
    print("求得的权值:\n", W)

    for i in range(cls):  # 数据降维
        Iris[i] = np.dot(Iris[i], W)  # 注意 shape 为(150, 2) X 是行向量

    mean = np.zeros((3, 2))
    for i in range(cls):
        mean[i] = np.mean(Iris[i], axis=0)

    acc = 0
    for i in range(cls):
        test[i] = np.dot(test[i], W)
        for p in test[i]:
            dis = []
            for j in range(cls):
                dis.append(np.sum((p - mean[j]) * (p - mean[j])))
            mx = dis.index(min(dis))
            if mx == i:
                acc += 1

    acc = acc / (cls * len(test[0]))
```

```
    plt.figure(figsize=(4, 4))
    col = ["r", "g", "b"]
    for i in range(cls):
        px = list()
        py = list()
        for p in Iris[i]:
            px.append(p[0])
            py.append(p[1])
        plt.scatter(px, py, c=col[i])

    plt.title('my LDA')
    plt.xlabel('LD1')
    plt.ylabel('LD2')
    plt.legend(labels, loc='best', fancybox=True)
    plt.show()

    return acc


if __name__ == "__main__":
    df = pd.read_csv("iris.data", header=None)
    Iris = list()
    labels = list()
    cls = 3  # 3类
    dim = 4  # 4维

    for i in range(cls):
        Iris.append(np.array(df.values[i * 50: (i + 1) * 50, 0:4],
dtype='float'))
        labels.append(df.values[i * 50, 4])
    print(labels)

    for i in range(cls):  # 随机打乱
        lis = [i for i in range(50)]
        random.shuffle(lis)
        Iris[i] = [Iris[i][j] for j in lis]

    siz = 40  # 训练集测试集 8 : 2
    train = list()
    test = list()
    for i in range(cls):
        train.append(Iris[i][:siz])
        test.append(Iris[i][siz:50])
```

```
acc = work(train, test)
print("acc: ", acc * 100, "%")
```