**人工智能实验：Topic 4——降维技术**

**Part 2: 线性判别分析**

**3190102060 黄嘉欣**

1. **LDA概述**

线性判别分析技术的主要思想是“投影后类内方差最小，类间方差最大”，即将数据在低维度上进行投影，投影后希望每一种类别数据的投影点尽可能的接近，而不同类别的数据的类别中心之间的距离尽可能的大。对两类二维数据而言，定义类内散度矩阵，类间散度矩阵为，其中表示第类样本的协方差矩阵，表示第类样本的均值向量。则为了求解投影到的最佳一维直线——最佳向量，问题等效为为何值时，广义瑞利商：取得最大值。由于的分子分母都是的二次项，因此解与的长度无关，只与其方向相关。其最大化问题等效为，。由拉格朗日乘子法，该式等价于，其中为拉格朗日乘子。由于的方向恒为，不妨令，最终可得。对多分类问题，若存在个类，则上述表达式修改为：，，其中为第类示例的协方差矩阵，为第类示例数，为类示例的均值向量，为整体数据的均值向量。由上式可以发现，的解为的前个非零特征值所对应的特征向量组成的矩阵，其所能降低的维度最大值为。

**二、LDA与PCA**

LDA与PCA均可以对数据进行降维，且两者在降维时均使用了矩阵特征分解的思想。然而，PCA是为了让映射后的样本具有最大的发散性；而LDA是为了让映射后的样本有最好的分类性能。因此，PCA是一种无监督的降维方法，LDA是一种有监督的降维方法。LDA降维最多能够使类数据的维数降低，但PCA没有这个限制；LDA除了可以用于降维，还可以用于分类。

**三、实验4-3**

① 实验题目

利用np.random.random()函数，生成两个类别的随机数据，样本大小为302（行表示样本数，2表示特征数），其中随机数A的取值范围为10-13，随机数据B的取值范围为15-18；通过LDA对生成的随机数据进行降维，并在同一张图内可视化降维直线和原始数据。具体的实现步骤为：① 定义函数——计算类内离散度矩阵；② 定义函数——计算类间离散度矩阵；③ 定义函数——LDA求解投影矩阵。

② 实验结果与分析

如图2.1，随机生成的A类样本取值范围为，B类样本的取值范围为。

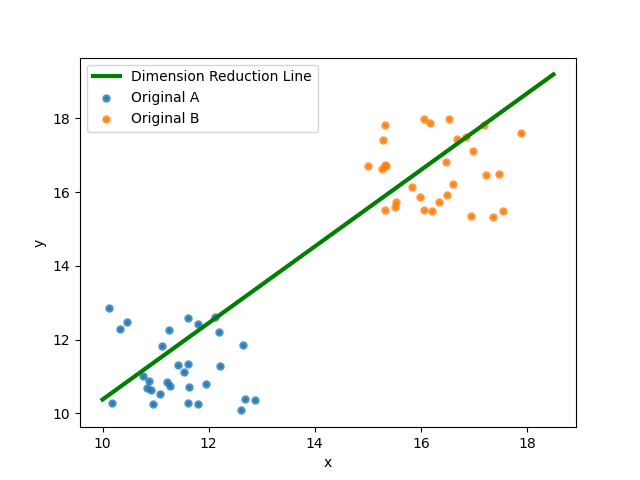


图2.1 降维直线与原始数据

与此同时，利用LDA算法得到的降维直线方程为，程序输出如下所示。



由图2.1可以发现，当A、B两类数据投影到降维直线上时，同类数据的投影点相对距离更近，不同类别数据中心之间的距离很大。因此，通过LDA，映射后的样本具有比较优秀的分类性能，实验效果符合预期。

**四、附录**：实验Python代码——*LDA.py*

from numpy import \*

import matplotlib.pyplot as plt

def compute\_Sw(A,B):

    """计算类内散度矩阵

    Args:

      A: A类原始数据

      B: B类原始数据

    Returns:

      Sw: 类内散度矩阵

    """

    meanA = mean(A,axis=0)             # A类均值

    meanB = mean(B,axis=0)             # B类均值

    SwA = dot((A-meanA).T,A-meanA)     # A类散度矩阵

    SwB = dot((B-meanB).T,B-meanB)     # B类散度矩阵

    Sw = SwA + SwB                     # 类内散度矩阵

    return Sw

def compute\_Sb(A,B):

    """计算类间散度矩阵

    Args:

      A: A类原始数据

      B: B类原始数据

    Returns:

      Sw: 类间散度矩阵

    """

    meanA = mean(A,axis=0)                              # A类均值

    meanB = mean(B,axis=0)                              # B类均值

    colVec = (meanA-meanB).reshape(len(meanA-meanB),-1) # 列向量

    rowVec = (meanA-meanB).reshape(-1,len(meanA-meanB)) # 行向量

    Sb = dot(colVec,rowVec)                             # 类间散度矩阵

    return Sb

def lda(A,B):

    """计算投影矩阵

    Args:

      A: A类原始数据

      B: B类原始数据

    Returns:

      W: 投影矩阵

    """

    Sw = compute\_Sw(A,B)                    # 计算类内散度矩阵

    Sb = compute\_Sb(A,B)                    # 计算类间散度矩阵

    mat = dot(linalg.inv(Sw),Sb)            # (Sw^-1)Sb

    eigenVals, eigenVecs = linalg.eig(mat) # (Sw^-1)Sb的特征值和特征矩阵

    maxEigenVal = argmin(eigenVals)         # 取第一个特征值

    W = eigenVecs[maxEigenVal]             # 第一个特征值对应的特征向量

    return W

def main():

    A = 10+3\*random.random((30,2))     # A的取值范围是10-13

    B = 15+3\*random.random((30,2))     # B的取值范围是15-18

    W = lda(A,B)                        # 投影矩阵

    data = append(A,B,axis=0)          # 整体样本

    lowDData = dot(data,W)             # 降维后的数据

    x = linspace(10,18.5,1500)

    y = W[1]/W[0]\*x

    print("The projected line is y=%fx" %(W[1]/W[0]))

    l1, = plt.plot(x,y,linewidth=3,color="green")         # 投影直线

    l2 = plt.scatter(A[:,0].tolist(),A[:,1].tolist(),marker='.',

                   cmap='Blues',alpha=0.8,linewidths=3)   # A类原始数据

    l3 = plt.scatter(B[:,0].tolist(),B[:,1].tolist(),marker='.',

                   cmap='Blues',alpha=0.8,linewidths=3)   # B类原始数据

    plt.legend(handles=[l1,l2,l3],labels=["Dimension Reduction Line",

               "Original A","Original B"],loc='best')

    plt.xlabel("x")

    plt.ylabel("y")

    plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    main()