人工智能实验：Topic 4——降维技术

Part 2: 线性判别分析

3190102060　黄嘉欣

一、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$\boldsymbol{S}_\omega = \boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1$，类间散度矩阵为$\boldsymbol{S}_b = (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^T$，其中$\boldsymbol{\Sigma}_i$表示第$i$类样本的协方差矩阵，$\boldsymbol{\mu}_i$表示第$i$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$\boldsymbol{\omega}$，问题等效为$\boldsymbol{\omega}$为何值时，广义瑞利商：$J(\boldsymbol{\omega}) = \frac{\boldsymbol{\omega}^T \boldsymbol{S}_b \boldsymbol{\omega}}{\boldsymbol{\omega}^T \boldsymbol{S}_\omega \boldsymbol{\omega}}$取得最大值。由于$J(\boldsymbol{\omega})$的分子分母都是$\boldsymbol{\omega}$的二次项，因此$J(\boldsymbol{\omega})$解与$\boldsymbol{\omega}$的长度无关，只与其方向相关。其最大化问题等效为$min - \boldsymbol{\omega}^T \boldsymbol{S}_b \boldsymbol{\omega}$，$s.t. \boldsymbol{\omega}^T \boldsymbol{S}_\omega \boldsymbol{\omega} = 1$。由拉格朗日乘子法，该式等价于$\boldsymbol{S}_b \boldsymbol{\omega} = \lambda \boldsymbol{S}_\omega \boldsymbol{\omega}$，其中$\lambda$为拉格朗日乘子。由于$\boldsymbol{S}_b \boldsymbol{\omega}$的方向恒为$\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1$，不妨令$\boldsymbol{S}_b \boldsymbol{\omega} = \lambda(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)$，最终可得$\boldsymbol{\omega} = \boldsymbol{S}_\omega^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)$。对多分类问题，若存在$N$个类，则上述表达式修改为：$\boldsymbol{S}_\omega = \sum_{i=1}^{N} \boldsymbol{S}_{\omega i}$，$\boldsymbol{S}_b = \sum_{i=1}^{N} m_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$，$\boldsymbol{S}_b \boldsymbol{W} = \boldsymbol{S}_\omega \boldsymbol{W}$，其中$\boldsymbol{S}_{\omega i}$为第$i$类示例的协方差矩阵，$m_i$为第$i$类示例数，$\boldsymbol{\mu}_i$为$i$类示例的均值向量，$\boldsymbol{\mu}$为整体数据的均值向量。由上式可以发现，$\boldsymbol{W}$的解为$\boldsymbol{S}_\omega^{-1} \boldsymbol{S}_b$的前$d$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d最大值为$N-1$。

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N类数据的维数降低$N-1$，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 30×2（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$\boldsymbol{S}_{\omega}$；② 定义函数——计算类间离散度矩阵$\boldsymbol{S}_b$；③ 定义函数——LDA 求解投影矩阵$\boldsymbol{W}$。

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[10,13]，B 类样本的取值范围为[15,18]。

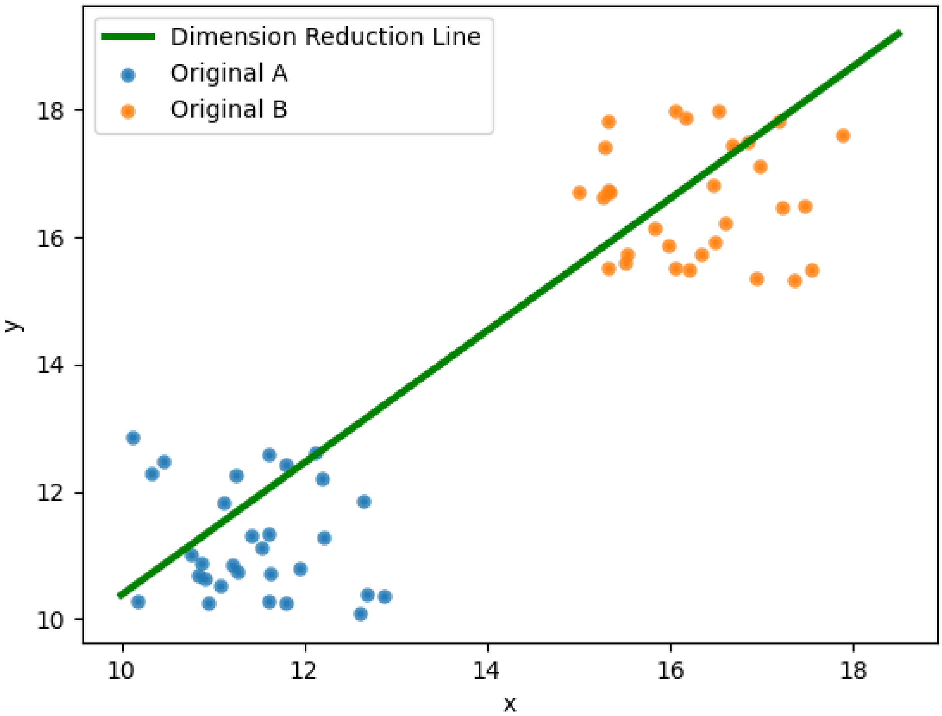

图 2.1 降维直线与原始数据

与此同时，利用 LDA 算法得到的降维直线方程为 $y = 1.037426x$，程序输出如下所示。

```
The projected line is y=1.037426x
```

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
```

```python
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
```

```python
    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()
```