



清华大学
Tsinghua University

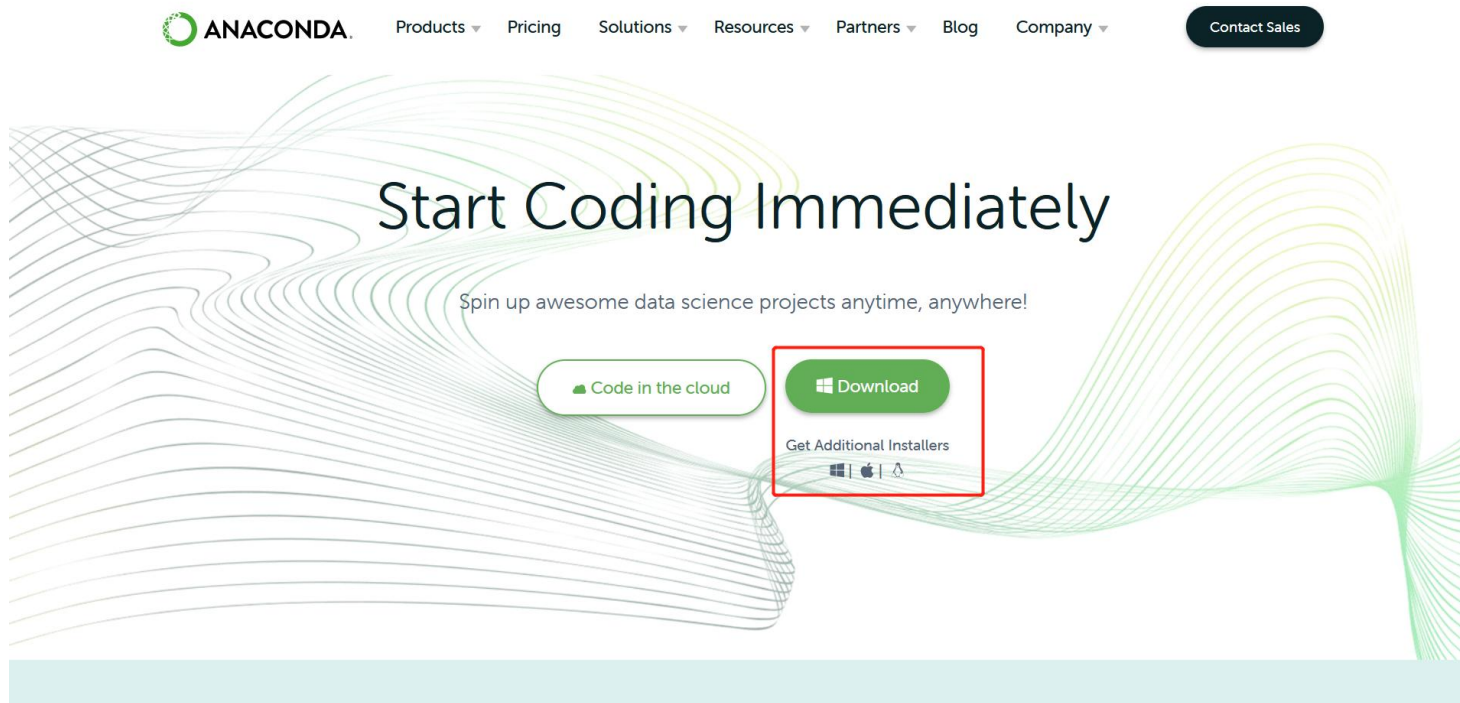
Fisher上机作业讲解

报告人：张济川

报告时间：2023-3-23

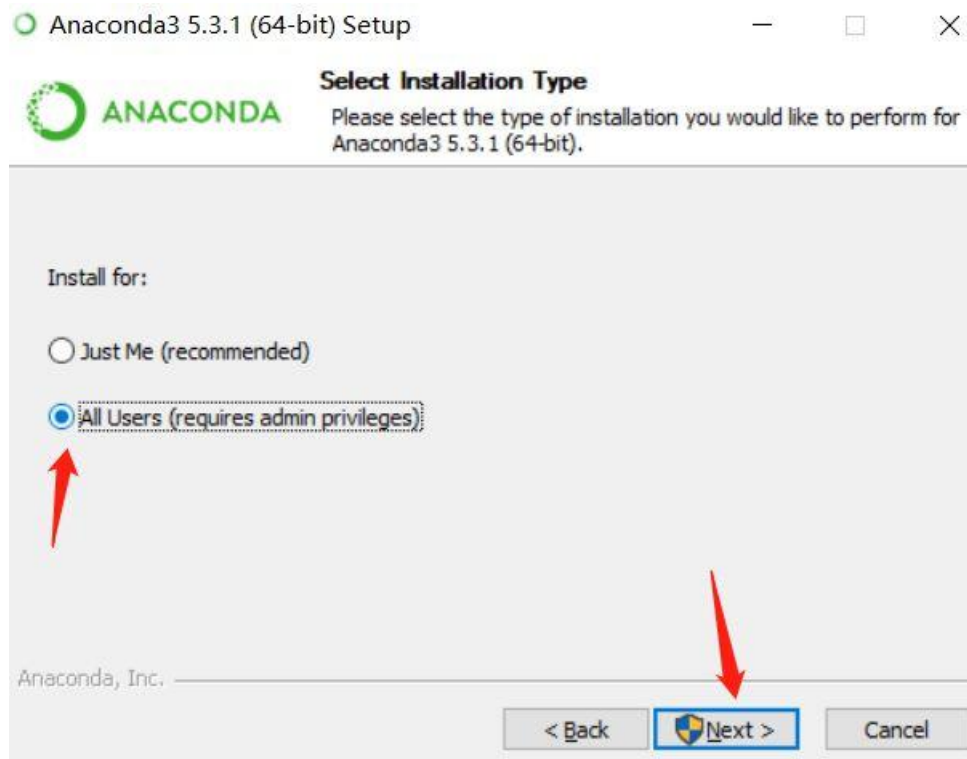
1 环境配置

- 在 <https://www.anaconda.com/> 下载 anaconda



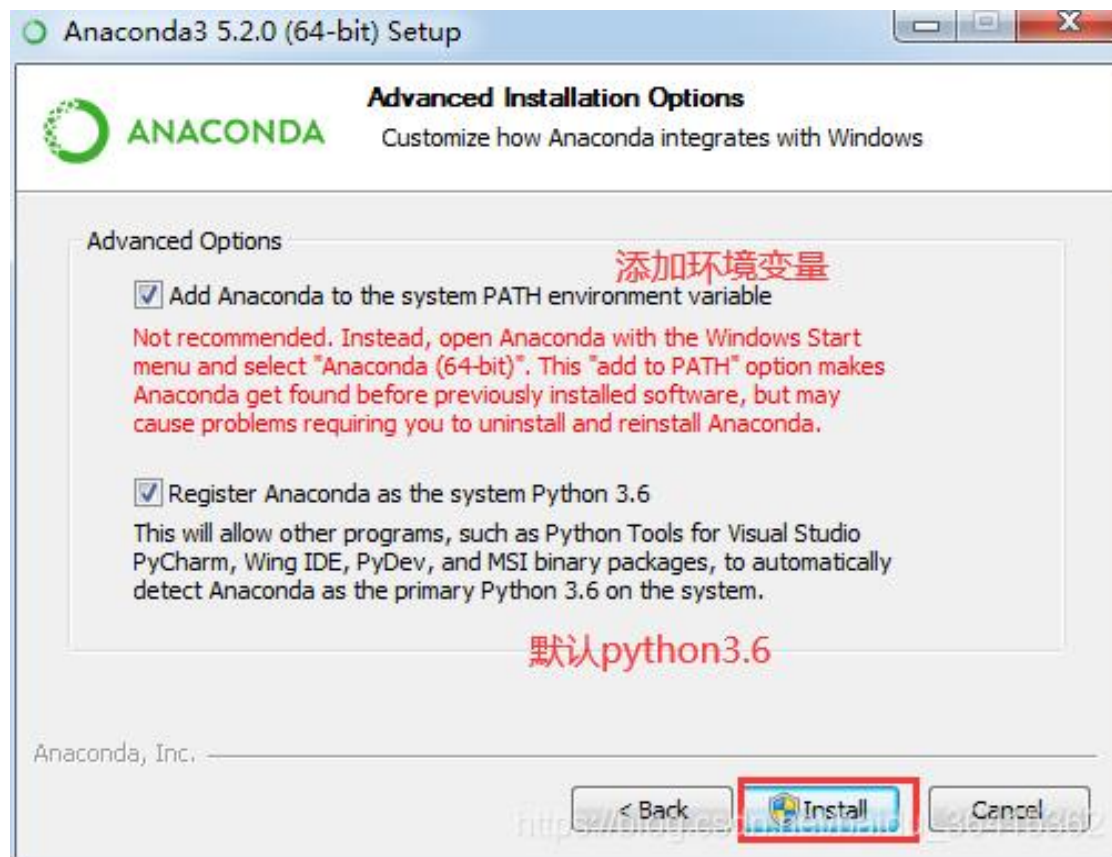
1 环境配置

- 安装设置



1 环境配置

- 安装设置



1 环境配置

- 检查是否安装成功

```
C:\Users\asus>conda --version  
conda 4.7.10  
  
C:\Users\asus>
```

- 换源

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/msys2/  
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/  
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/  
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch/  
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/  
conda config --set show_channel_urls yes
```

- 查看

```
conda config --show
```



1 环境配置

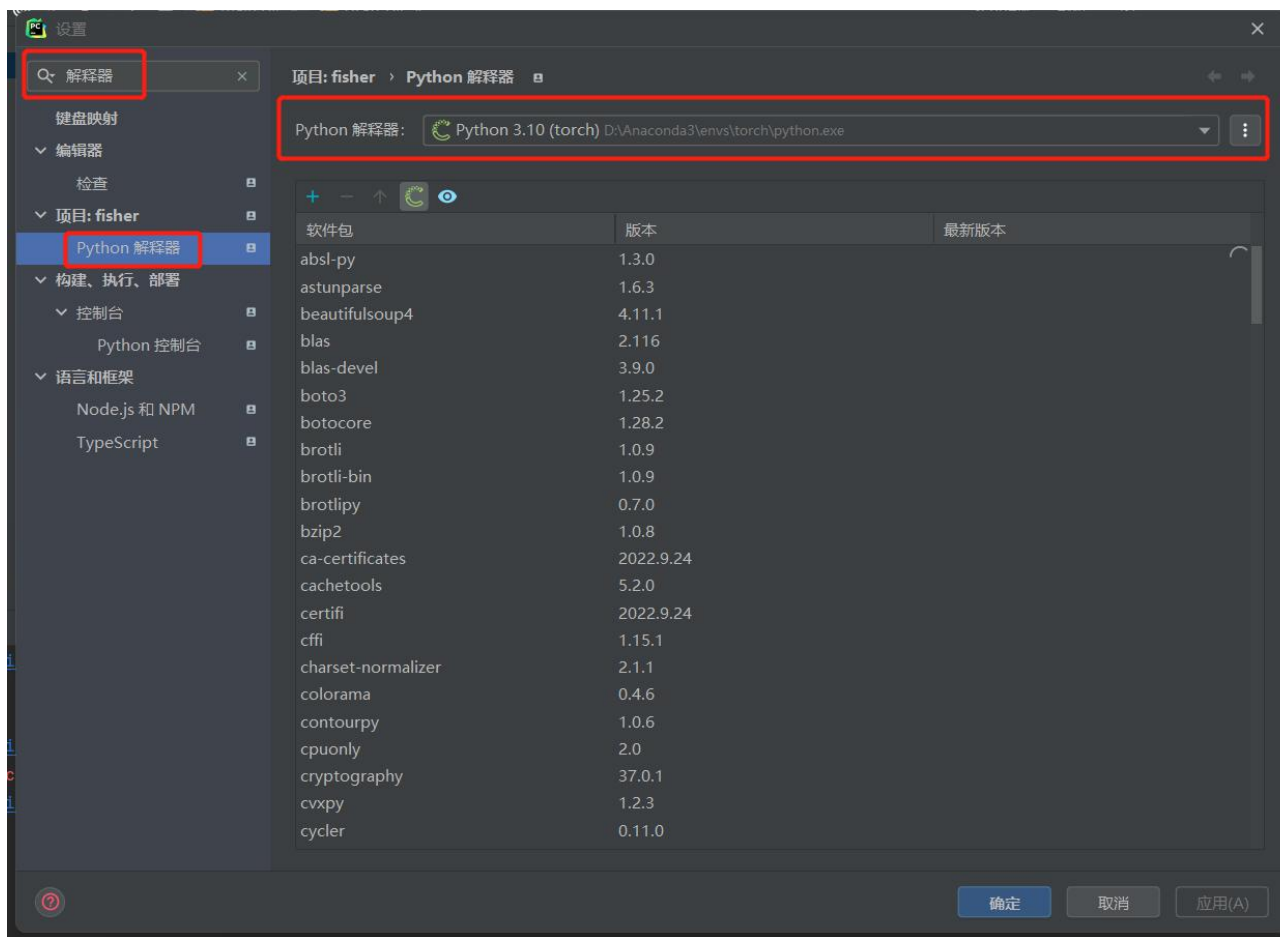
- 在命令行输入: `conda create -n media python=3.7` // 创建一个名为media的环境并指定python版本为3.7
- `conda activate media` // 进入media环境
- `pip install numpy matplotlib` // 安装需要的库
- `conda deactivate` // 退出media环境

```
(media) C:\Users\zhang>python
Python 3.7.12 | packaged by conda-forge | (default, Oct 26 2021, 05:35:01) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> import matplotlib
>>> exit()
```



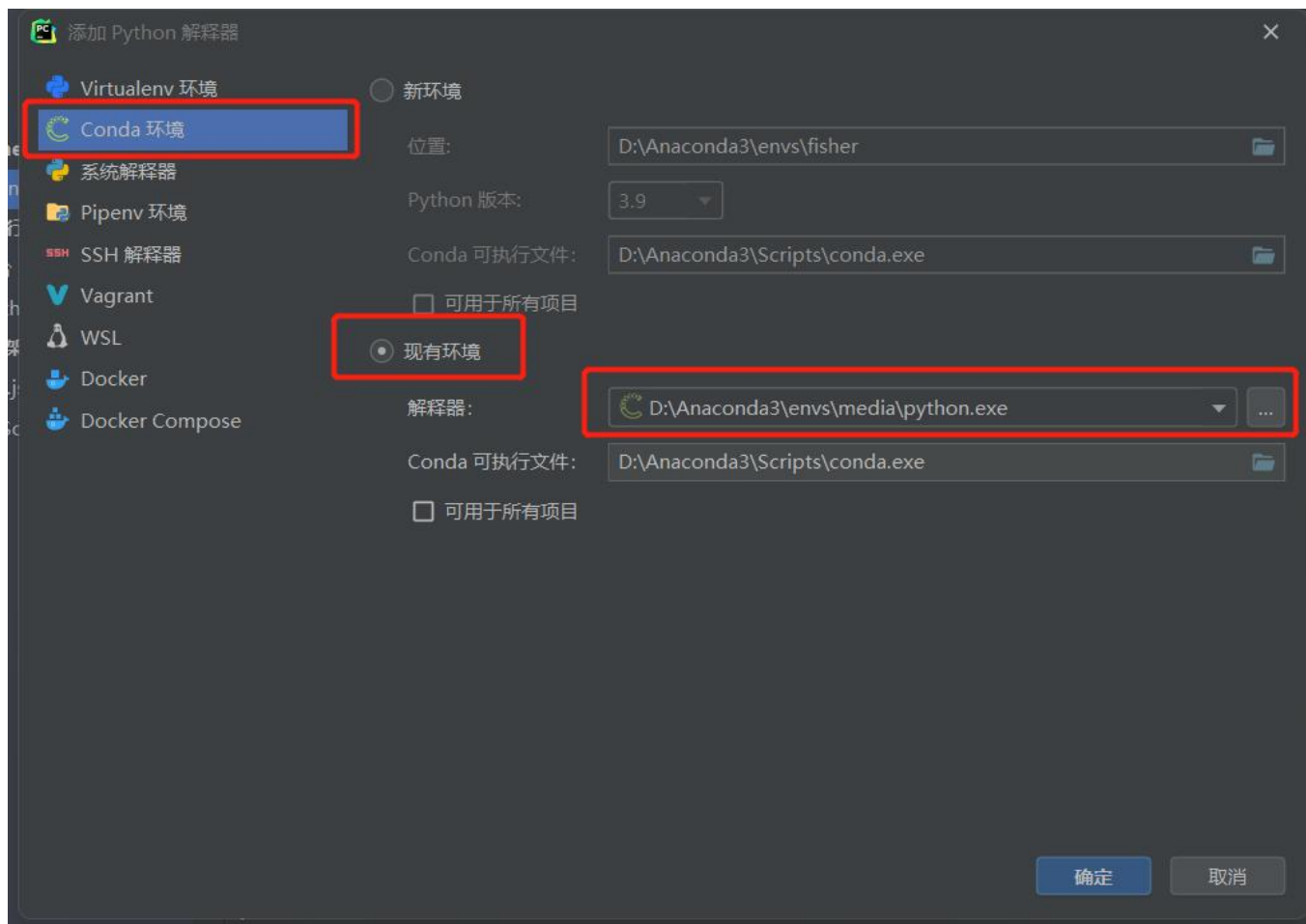
1 环境配置

- 用PyCharm或者vscode打开文件夹，并设置python解释器



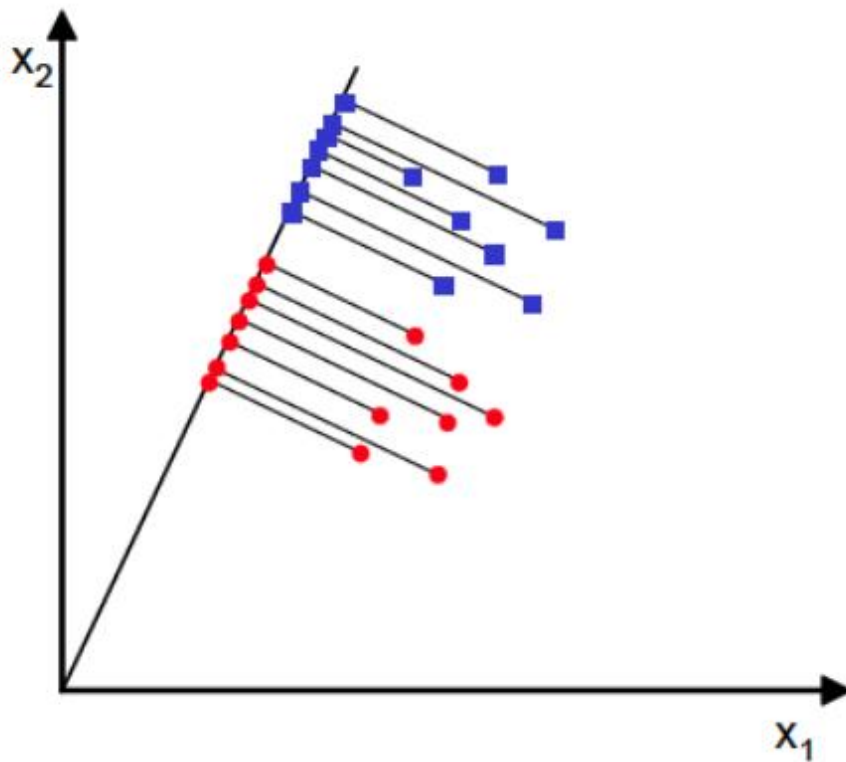
1 环境配置

- 用PyCharm或者vscode打开文件夹，并设置python解释器



2 理论回顾

- Fisher算法目标：寻找投影向量和分类阈值，完成二分类任务。



2 理论回顾

- Fisher算法流程

Fisher线性判别算法:

给定两类数据 w_1 与 w_2 , 本次实验中, $w_1: (100, 2)$, $w_2: (100, 2)$ 。

1. 求两类样本均值向量 u_1 和 u_2
2. 求两类样本类内离散度矩阵 $S_i = \sum_{y \in w_i} (x - u_i)(x - u_i)^T, i = 1, 2$
3. 总的类内散度矩阵 $S_w = S_1 + S_2$
4. **最优投影向量** $w = S_w^{-1}(u_1 - u_2)$
5. 对两类已知样本, 求出它们在 w 上的投影点 $Y_i = w^T X_i$
6. 求各类样本投影后的均值 $\overline{Y_i}$
7. **分类阈值** $w_0 = \frac{n_1}{n_1+n_2} \overline{Y_1} + \frac{n_2}{n_1+n_2} \overline{Y_2}$
8. 对未知样本 X , 计算投影点 Y , 根据判别规则对未知样本 X 进行分类, 若 $Y > w_0$, 则 $X \in w_1$; 若 $Y < w_0$, 则 $X \in w_2$ 。



3 实验

- 题目描述

2. (编程题) Fisher 线性判别

本题中，我们通过编程实现 Fisher 线性判别：

(1) 分别为两个类别 ω_1 和 ω_2 各生成 100 个样本点，并画出生成的样本点。 ω_1 和 ω_2 服从高斯分布，高斯分布的均值和方差分别为：

$$\mu_1 = \begin{bmatrix} 1 \\ 5 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mu_2 = \begin{bmatrix} 5 \\ -3 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

(2) 计算 Fisher 线性判别的投影向量 ω 和投影阈值 ω_0 ，并在图中画出直线 $y = \omega^T x$

(3) 计算最终 Fisher 线性判别的准确率



3 实验

- ChatGPT

python怎么根据均值和协方差矩阵生成二维高斯分布



要根据均值和协方差矩阵生成二维高斯分布，可以使用NumPy库的随机数模块`numpy.random`中的`multivariate_normal`函数。

以下是一个示例代码，其中假设均值为`mu`，协方差矩阵为`cov`，并生成1000个随机样本：

```
python Copy code

import numpy as np

# 定义均值和协方差矩阵
mu = np.array([1, 2])
cov = np.array([[1, 0.5], [0.5, 2]])

# 使用multivariate_normal函数生成随机样本
samples = np.random.multivariate_normal(mu, cov, 1000)

# 打印前10个样本
print(samples[:10])
```

输出结果类似于：

```
css Copy code

[[ 2.20663612  2.99264561]
 [ 1.22809848  1.80158254]
 [-0.07048618  0.59511736]
 [ 1.4085736   2.11467609]
 [ 1.31056967  3.53340658]
 [ 2.32406336  1.33100064]
 [ 0.52867446  0.81343783]
 [ 1.14708211  1.5143542 ]
 [ 0.53135691  0.17410194]
 [ 1.73287906  1.75849628]]
```

其中每一行代表一个二维高斯分布的样本，第一列是x坐标，第二列是y坐标。



3 实验

- step 1: 数据生成与可视化

```
# 使用multivariate_normal函数生成随机样本
w1 = np.random.multivariate_normal(mu1, cov1, 100) # ndarray:(100,2)
```

```
# 绘制散点图，用不同颜色表示两类数据
plt.scatter(w1[:, 0], w1[:, 1], color='red', label="w1")
plt.scatter(w2[:, 0], w2[:, 1], color='blue', label="w2")
```

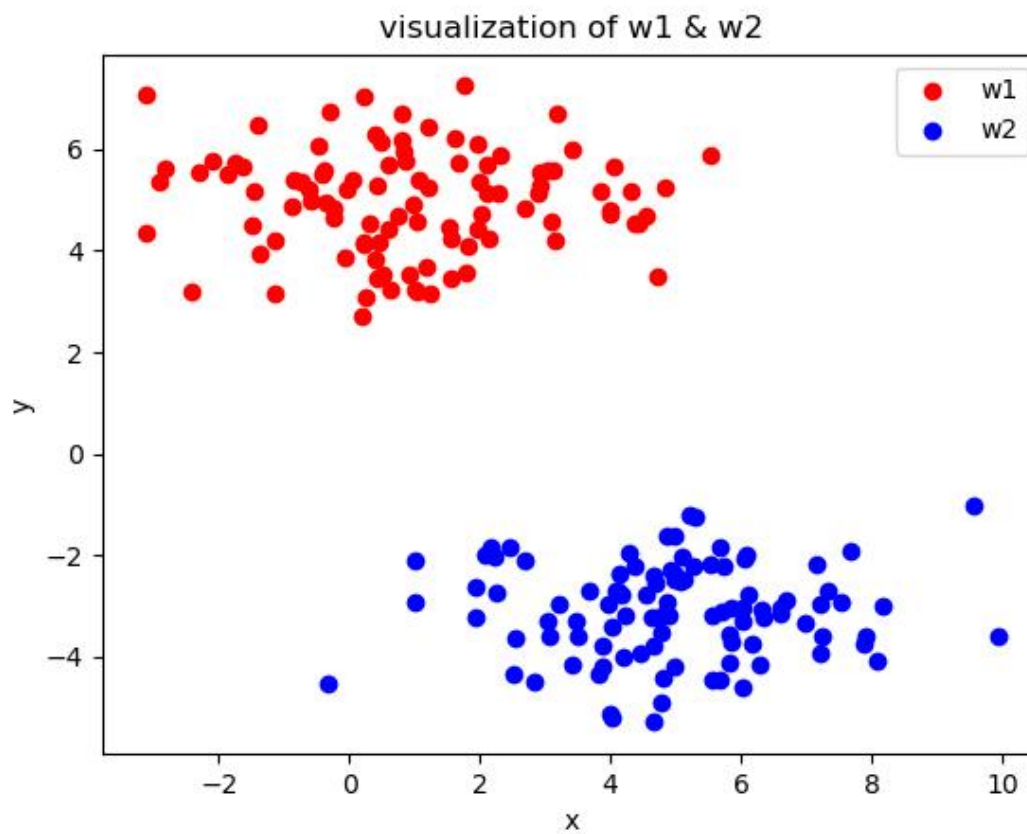
```
# 添加标题和标签
plt.title('visualization of w1 & w2')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
```

```
# 显示图像
plt.show()
```



3 实验

- 数据可视化效果图



3 实验

- step 2: 计算类内散度矩阵

1. 求两类样本均值向量 u_1 和 u_2
2. 求两类样本类内离散度矩阵 $S_i = \sum_{y \in w_i} (x - u_i)(x - u_i)^T, i = 1, 2$
3. 总的类内散度矩阵 $S_w = S_1 + S_2$

```
# 求w1、w2的均值，可以用np.mean()函数，注意是在axis=0这个维度上取平均，返回一个大小为(2,)的
ndarray
w1_mean = ...
w2_mean = ...

# 计算S1
# 方法一：循环
# np.zeros((2,2))可以生成一个二维全零矩阵
# np.expand_dims(..., axis=1)可以将shape为(?,)的ndarray变为(?,1)，便于做矩阵乘法
# @:矩阵乘法，.T:转置
# 方法二：矩阵乘法
# .T:转置
# np.dot(A,B)可以实现矩阵乘法AB
# 方法三：np.cov函数
# np.cov函数返回协方差矩阵，计算的是各个特征之间的协方差
...

# 计算S2
...

# 计算Sw
Sw = ...
```



3 实验

- step 3: 计算最优投影向量及可视化

4. 最优投影向量 $w = S_w^{-1}(u_1 - u_2)$

```
# 计算最优投影向量w
# 求逆: np.linalg.inv()
# np.dot(A,x)可以实现矩阵A与向量x的乘法:Ax, 等价于A.dot(x)
w = ...

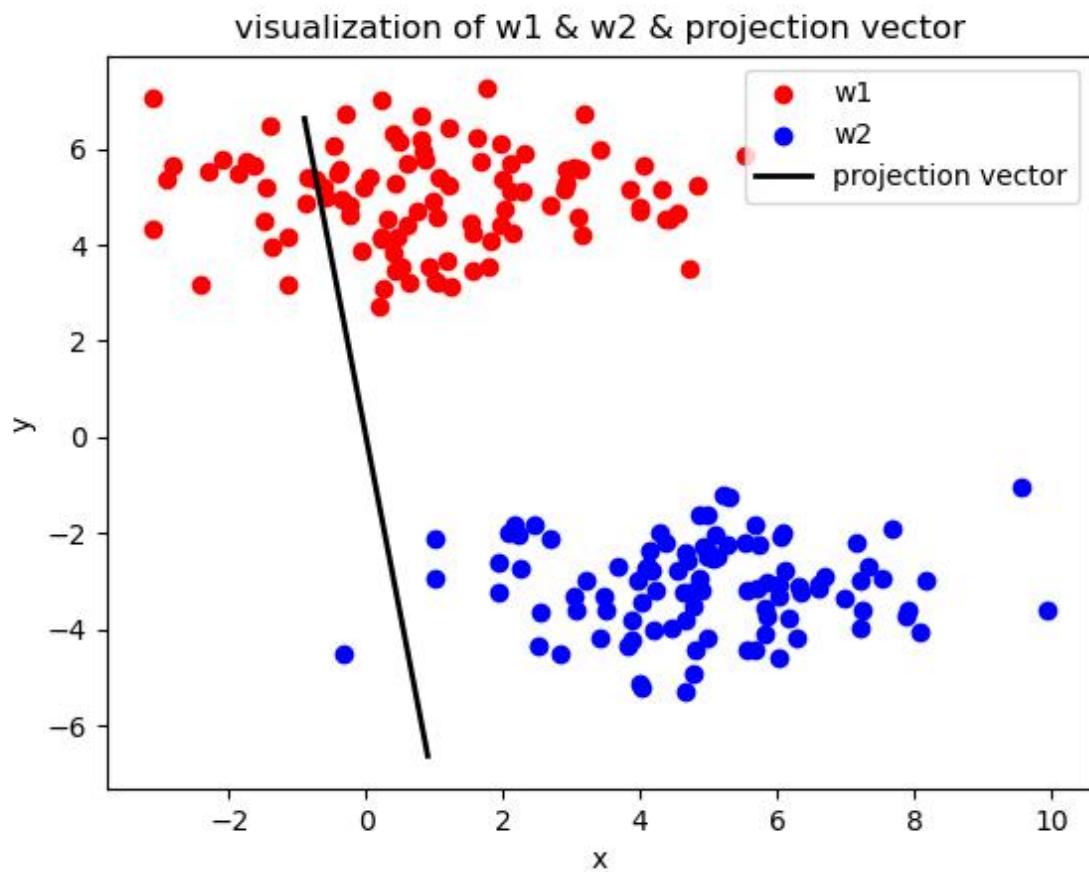
# 对w代表的直线做可视化
x = np.linspace(start=-0.9, stop=0.9, num=100)
y = (w[1] * x) / w[0]
plt.plot(x, y, linewidth='2', label="projection vector", color='black')

plt.title('visualization of w1 & w2 & projection vector')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
# 显示图像
plt.show()
```



3 实验

- 最优投影向量可视化效果图



3 实验

- step 4: 计算分类阈值及可视化

5. 对两类已知样本, 求出它们在 w 上的投影点 $Y_i = w^T X_i$

6. 求各类样本投影后的均值 $\overline{Y_i}$

7. **分类阈值** $w_0 = \frac{n_1}{n_1+n_2} \overline{Y_1} + \frac{n_2}{n_1+n_2} \overline{Y_2}$

```
# 计算分类阈值
# np.dot()与np.mean()
w0 = ...
```

```
plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定默认字体
plt.rcParams['axes.unicode_minus'] = False # 解决保存图像是负号 '-' 显示为方块的问题

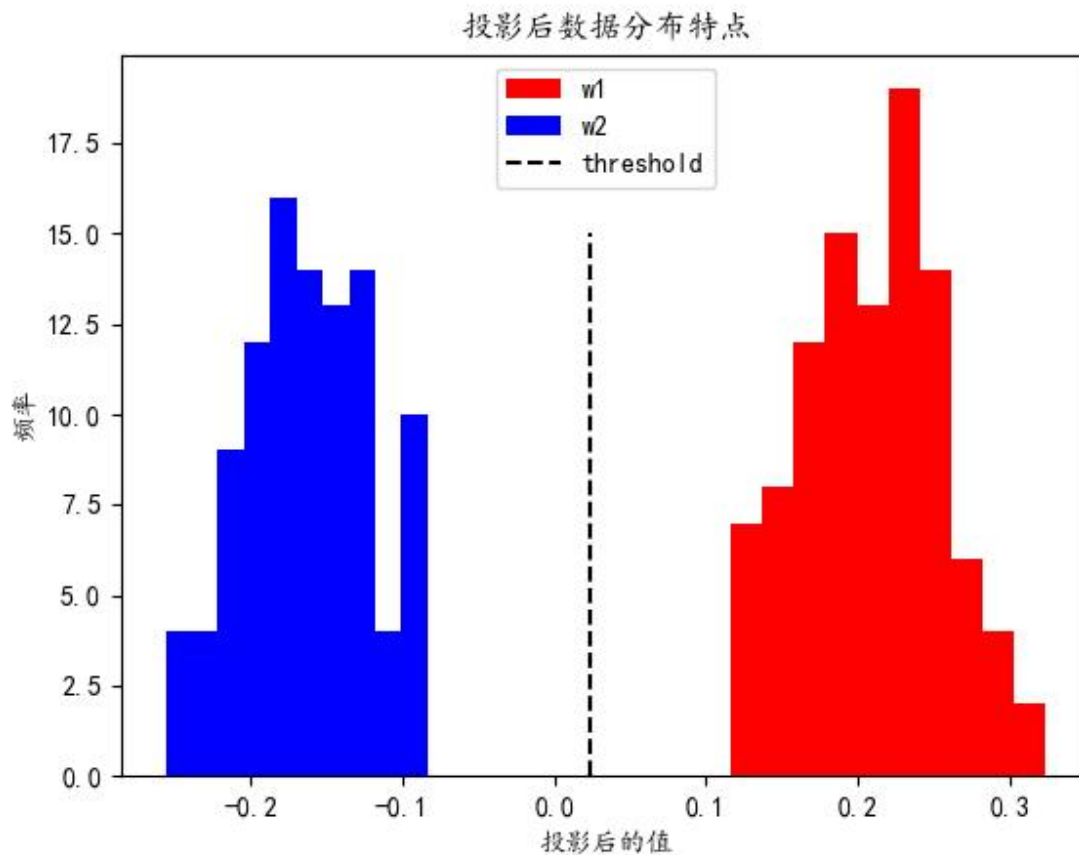
plt.hist(w1.dot(w), color="red", label="w1")
plt.hist(w2.dot(w), color="blue", label="w2")
plt.vlines([w0], 0, 15, linestyle='dashed', colors='black', label="threshold")
plt.legend(loc='upper center')

...
plt.show()
```



3 实验

- 分类阈值可视化效果图



3 实验

- step 5: 计算分类准确率

若 $Y > w_0$, 则 $X \in w_1$; 若 $Y < w_0$, 则 $X \in w_2$ 。

```
w1_pred_correct_num = np.count_nonzero(w1.dot(w) > w0)
# w1.dot(w)是投影结果, 大小为(100,)
# w1.dot(w) > w0返回一个大小为(100,)的布尔矩阵
# np.count_nonzero用于计算一个ndarray中True的个数
# w1_pred_correct_num: w1中被正确分类的元素数量

w2_pred_correct_num = np.count_nonzero(w2.dot(w) < w0)
# w2_pred_correct_num: w2中被正确分类的元素数量

# 计算分类准确率
accuracy = (w1_pred_correct_num + w2_pred_correct_num) / (w1.shape[0] +
w2.shape[0])
```

分类准确率: 1.0



谢谢大家！欢迎批评指正！

