Fisher线性判别

环境

- pycharm专业版2019.3.3
- python3.7.4
- 外部库：numpy=1.16.5；sklearn=0.21.3；matplotlib=3.3.1；openpyxl=3.0.0

原理简介

1. 将高纬度问题降维，即假设数据存在于n维空间中，在数学上，通过投影使数据到一条直线上。然后根据投影点在直线上的分布对原始点进行分类。
2. 怎么找到合适的直线方向，能使不同类别数据映射到该条直线上易于区分，这就是Fisher线性判别要解决的问题。

一些计算公式

在n维X空间

- 各类样本均值向量μ_i:

$$\mu_i = \frac{1}{N_i}\sum_{x_j \in \Omega_i} x_j, i = 1,2 \tag{1}$$

- 各类类内离散度矩阵S_i:

$$S_i = \sum_{x_j \in \Omega_i} (x_j - \mu_i)(x_j - \mu_i)^T, i = 1,2 \tag{2}$$

- 总类内离散度矩阵S_w:

$$S_w = \sum_{x_j \in \Omega_i} S_i, i = 1,2 \tag{3}$$

- 样本类间离散度矩阵S_b:

$$S_b = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \tag{4}$$

在1维Y空间

- 各类样本均值$\overline{\mu_i}$:

$$\overline{\mu_i} = \frac{1}{N_i}\sum_{y_j \in \psi_i} y_j, i = 1,2 \tag{5}$$

- 各类内离散度$\overline{S_i^2}$:

$$\overline{S_i}^2 = \sum_{y_j \in \psi_i} (y_j - \overline{\mu}_i)^2, i = 1,2 \tag{6}$$

- 最佳投影方向ω^*:

$$\omega^* = {S_\omega}^{-1}(\mu_1 - \mu_2) \tag{7}$$

- 决策点y_0:

$$y_0 = \frac{\overline{\mu}_1 + \overline{\mu}_2}{2} \tag{8}$$

分类评价指标

- 总体分类精度OA：

$$\frac{\text{所有判断正确的样本数}}{\text{所有测试样本数}} \times 100\% \tag{9}$$

- 类别分类精度AA:

$$\frac{\text{某一类中判断正确的样本数}}{\text{该类参与测试的样本数}} \times 100\% \tag{10}$$

- kappa系数:

$$\frac{OA - pe}{1 - pe} \tag{11}$$

- pe:

$$\frac{\sum\left(\text{某类参与测试的样本数} \times \text{被判断为该类的测试样本数}\right)}{\text{所有的测试样本数}^2} \tag{12}$$

数据标准化

由于两种数据集中特征的量纲都一样，所以为了消除量纲，将每个特征的数值除以该特征中的最大值即可，这样将所有特征数值映射到了区间[0,1]之间

划分训练集和测试集

利用sklearn包中的train_test_split函数进行训练集和测试集的划分，其中训练集占比40%，测试集占比60%。

其中train_test_split函数的一般形式如下：

```
X_train,X_test, y_train, y_test
=train_test_split(train_data,train_target,test_size=0.4, random_state=0)
```

参数解释：

- **train_data**：所要划分的样本特征集
- **train_target**：所要划分的样本结果集
- **test_size**：测试样本占比，如果是整数的话就是样本的数量
- **random_state**：是随机数的种子
- **随机种子**：其实就是该组随机数的编号，在需要重复试验的时候，保证得到一组一样的随机数

通过该函数，只要每次传入不同的随机种子，就可以得到不同的训练集、测试集

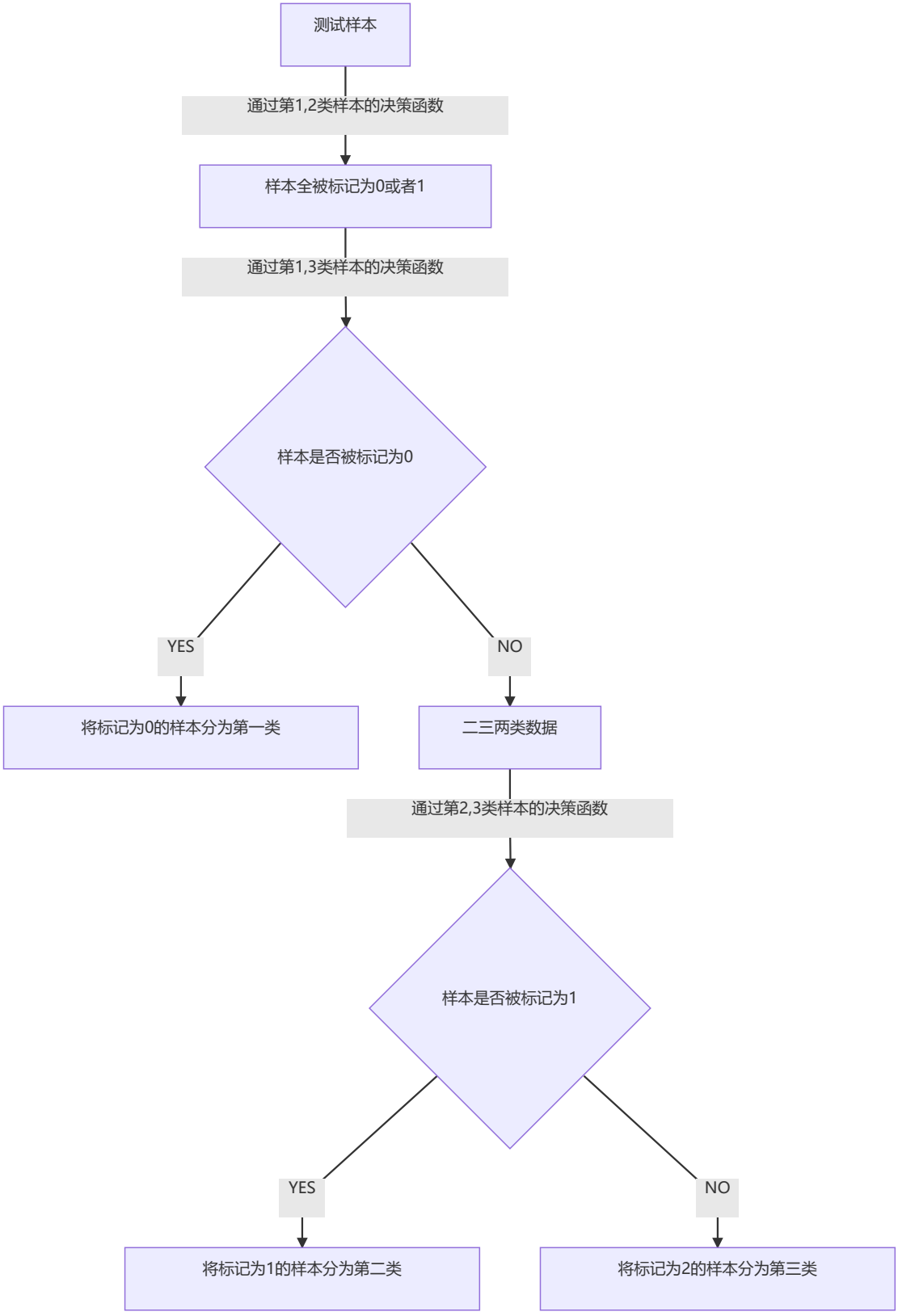
分类策略

1. sonar数据集

   sonar数据集为一个二分类问题，只需要计算出最优投影方向，投影后计算出决策点，再进行分类即可

2. iris数据集

   iris数据集为一个三分类问题，可以将其转化为三个二分类问题进行分类，具体分类思路如下：



分类结果

1. sonar数据集

试验次数	整体分类精度 OA	kappa系数	第一类类别分类精度 AA	第二类类别分类精度 AA
1	0.603	0.213	0.678	0.537
2	0.690	0.370	0.559	0.806
3	0.690	0.380	0.695	0.687
4	0.587	0.171	0.559	0.612
5	0.690	0.385	0.763	0.627
6	0.698	0.389	0.610	0.776
7	0.675	0.355	0.763	0.597
8	0.683	0.371	0.780	0.597
9	0.706	0.410	0.678	0.731
10	0.690	0.383	0.729	0.657
11	0.643	0.284	0.627	0.657
12	0.730	0.467	0.847	0.627
13	0.643	0.279	0.576	0.701
14	0.683	0.353	0.542	0.806
15	0.675	0.351	0.712	0.642
16	0.603	0.198	0.525	0.672
17	0.794	0.589	0.847	0.746
18	0.611	0.223	0.627	0.597
19	0.627	0.246	0.542	0.701
20	0.778	0.551	0.712	0.836
21	0.675	0.358	0.797	0.567
22	0.683	0.365	0.695	0.672
23	0.754	0.500	0.644	0.851
24	0.738	0.475	0.729	0.746
25	0.730	0.460	0.746	0.716
26	0.667	0.335	0.695	0.642
27	0.667	0.333	0.678	0.657
28	0.706	0.417	0.780	0.642
29	0.754	0.508	0.780	0.731
30	0.730	0.462	0.780	0.687

试验次数	整体分类精度 OA	kappa系数	第一类类别分类精度 AA	第二类类别分类精度 AA
平均值	**0.687**	**0.373**	**0.690**	**0.684**

2. iris数据集

实验次数	整体分类精度OA	kappa系数	类别分类精度AA（依次为第1，2，3类）
1	0.900	0.850	1.00 0.767 0.933
2	0.911	0.867	1.00 0.800 0.933
3	0.833	0.750	1.00 0.800 0.700
4	0.956	0.933	1.00 0.900 0.967
5	0.922	0.883	1.00 0.900 0.867
6	0.967	0.950	1.00 0.900 1.000
7	0.900	0.850	1.00 0.800 0.900
8	0.944	0.917	1.00 0.900 0.933
9	0.933	0.900	1.00 0.967 0.833
10	0.956	0.933	1.00 0.933 0.933
11	0.900	0.850	1.00 0.833 0.867
12	0.822	0.733	1.00 0.667 0.800
13	0.833	0.75	1.00 0.800 0.700
14	0.967	0.950	1.00 0.933 0.967
15	0.878	0.817	1.00 0.633 1.000
16	0.922	0.883	1.00 0.933 0.833
17	0.644	0.467	1.00 0.433 0.500
18	0.900	0.850	1.00 0.833 0.867
19	0.900	0.850	1.00 0.867 0.833
20	0.889	0.833	1.00 0.867 0.800
21	0.900	0.850	1.00 0.900 0.800
22	0.900	0.850	1.00 0.833 0.867
23	0.911	0.867	1.00 0.967 0.767
24	0.944	0.917	1.00 0.833 1.000
25	0.944	0.917	1.00 0.967 0.867
26	0.922	0.883	0.97 0.867 0.933
27	0.956	0.933	1.00 0.900 0.967
28	0.922	0.883	1.00 0.900 0.867
29	0.956	0.933	1.00 0.933 0.933
30	0.878	0.817	1.00 0.767 0.867

实验次数	整体分类精度OA	kappa系数	类别分类精度AA（依次为第1，2，3类）
平均值	0.904	0.856	0.999　0.844　0.868

结果分析

1. sonar

   从结果可以看出，sonar数据集的分类精度并不理想，只有68.7%，kappa系数也只有0.373，这说明分类效果较为一般，个人认为分类效果一般的原因如下：

   sonar数据集的维度较高，足足有60维，而我们直接将其降到了一维，降维的过程中避免不了有效信息的损失，有可能是因为有效信息损失过多，导致分类效果不理想。

   可行的改进方法是不降成一维，降成2维或者稍低一点的维度进行分类。

2. iris数据集

   从结果可以看出，三分类的准确率达到了90.4%，kappa系数达到了0.856，说明分类效果很好，与实际情况几乎完全一致。

   类别分类精度中第一类的精度最高，二三类稍低一点，可以看出：第一类鸢尾花在四个特征上与另外两类有较为明显的差别，很容易跟另外两类区分开来；而第二三类可能是在四个特征上的差别没有特别明显，所以分类精度会有所下降。

代码展示

1. 鸢尾花数据集分类（数据来源于sklearn内部封装的数据集）

```python
# -*- coding: utf-8 -*-

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import numpy as np
from matplotlib import pyplot as plt
import random


# 计算并返回均值向量
def junzhi(iris):
    a = np.zeros([4, 1])
    a[0] = np.mean(iris[:, 0])
    a[1] = np.mean(iris[:, 1])
    a[2] = np.mean(iris[:, 2])
    a[3] = np.mean(iris[:, 3])
    return a

# 计算类内离散度矩阵S_i
def S_i(iris):
    a = junzhi(iris)
    b = np.zeros([iris.shape[1], iris.shape[1]])
    for i in range(iris.shape[0]):
        b = b + np.matmul((iris[i, :].T - a), (iris[i, :].T - a).T)
    return b

# 计算类间离散度矩阵S_b
def S_b(iris1, iris2):
    b_1 = S_i(iris1)
    b_2 = S_i(iris2)
```

```
    c = b_1 + b_2
    return c

# 划分训练集、测试集的函数
def train_test(iris, target, num):
    train_iris, test_iris, train_target, test_target =\
        train_test_split(iris, target, test_size=0.6, random_state=num,
shuffle=True)
    return {'train_iris': train_iris, 'test_iris': test_iris,
'train_target': train_target, 'test_target': test_target}

# 计算出最优的投影方向并计算出决策点
def best_w(iris1, iris2, target1, target2, num):
    train_iris1 = train_test(iris1, target1, num)['train_iris']
    train_iris2 = train_test(iris2, target2, num)['train_iris']
    s_0 = S_b(train_iris1, train_iris2)
    best_w = np.matmul(np.linalg.inv(s_0), junzhi(train_iris1) -
junzhi(train_iris2))
    y_0 = 0.5*np.mean(np.matmul(train_iris1, best_w)) +
0.5*np.mean(np.matmul(train_iris2, best_w))
    # print(best_w)
    return best_w, y_0

# 对测试样本进行分类并计算相关评价指标
def classify(iris1,iris2,iris3,target1,target2,target3,num):
    # print(num)
    ## 训练集得到的最佳方向和决策点
    w_best12,y0_12=best_w(iris1, iris2, target1, target2, num)
    w_best13,y0_13=best_w(iris1, iris3, target1, target3, num)
    w_best23,y0_23=best_w(iris2, iris3, target2, target3, num)
    ## 测试集
    test1=train_test(iris1,target1,num)['test_iris']
    test2=train_test(iris2,target2,num)['test_iris']
    test3=train_test(iris3,target3,num)['test_iris']
    test=np.vstack((test1,test2,test3))
    ## 当前测试集对应的标签
    test_target1=train_test(iris1,target1,num)['test_target']
    test_target2=train_test(iris2,target2,num)['test_target']
    test_target3=train_test(iris3,target3,num)['test_target']
    test_target=np.hstack((test_target1,test_target2,test_target3))
    # print(test_target)
    ## 存放预测得到的标签
    predict_target=np.zeros_like(test_target)
    ## 先通过第一二类决策函数
    y=np.matmul(test,w_best12)
    for i in range(len(test)):
        if y[i]>y0_12 or y[i]==y0_12:
            predict_target[i]=0
        else:
            predict_target[i]=1
    ## 再通过第一三类决策函数
    y=np.matmul(test,w_best13)
    for i in range(len(test)):
        if y[i]>y0_13 or y[i]==y0_13:
            predict_target[i]=0
        else:
            predict_target[i]=2

```

```
    ## 剩余的通过第二三类决策函数
    y=np.matmul(test,w_best23)
    for i in range(len(test)):
        if predict_target[i]!=0:
            if y[i]>y0_23 or y[i]==y0_23:
                predict_target[i]=1
            else:
                predict_target[i]=2
    # print(predict_target)
    ## 计算OA、AA、kappa系数
    ### 记录三类样本分类正确的数量
    num_1=0
    num_2=0
    num_3=0
    ### 记录三类样本实际分类的数量
    real_num_1=0
    real_num_2=0
    real_num_3=0
    for i in range(len(test_target)):
        ## 统计分类正确的数量
        if i<len(test_target)/3:
            if predict_target[i]==test_target[i]:
                num_1=num_1+1
        elif i<2*(len(test_target))/3:
            if predict_target[i]==test_target[i]:
                num_2=num_2+1
        else:
            if predict_target[i]==test_target[i]:
                num_3=num_3+1
        ## 统计实际的分类数量
        if predict_target[i]==0:
            real_num_1=real_num_1+1
        elif predict_target[i]==1:
            real_num_2=real_num_2+1
        else:
            real_num_3=real_num_3+1
    # print(num_1)
    ### 计算相关指标
    OA=(num_1+num_2+num_3)/len(test_target)
    AA_1=num_1*3/len(test_target)
    AA_2=num_2*3/len(test_target)
    AA_3=num_3*3/len(test_target)
    pe=(real_num_1*len(test_target1)+real_num_2*len(test_target2)\
        +real_num_3*len(test_target3))/np.square(len(test_target))
    kappa=(OA-pe)/(1-pe)
    return OA,[AA_1,AA_2,AA_3],kappa


if __name__ == "__main__":
    # 导入数据并去量纲
    data = load_iris()
    iris1 = data.data[0:50, 0:4]
    iris2 = data.data[50:100, 0:4]
    iris3 = data.data[100:150, 0:4]

    max_feature=np.array([np.max(data.data[:,0]),np.max(data.data[:,1]),\
        np.max(data.data[:,2]),np.max(data.data[:,3])])
    iris1=iris1/max_feature
```

```
    iris2=iris2/max_feature
    iris3=iris3/max_feature
    # 导入标签
    target1 = data.target[0:50].T
    target2 = data.target[50:100].T
    target3 = data.target[100:150].T

    # 存储相关的指标值
    OAs=np.zeros([30,1])
    AAs=np.zeros([30,3])
    kappas=np.zeros([30,1])

    #随机给出30个随机种子，用于train_test函数
    # nums=
[703,5205,8248,4998,1027,8528,7063,6513,793,2805,1524,8985,3939,9000\
    #
,3796,3178,628,9359,582,265,5920,8866,7960,5090,5481,4928,526,8763,5333,659
6]
    nums=random.sample(range(0,10000),100)
    j=0;k=0
    while j<30:
        try:

 OAs[j],AAs[j,],kappas[j]=classify(iris1,iris2,iris3,target1,target2,target
3,nums[k])
            j=j+1
            k=k+1
        except:
            k+=1
    # print(OAs)
    temp=0
    for i in range(len(OAs)):
        if OAs[i]!=0:
            temp=temp+1
    # print(AAs)
    # print(OAs)
    # print(kappas)
    print("AA值分别为:
\n{:.3f}\n{:.3f}\n{:.3f}".format(np.sum(AAs[:,0])/temp,np.sum(AAs[:,1]/temp
),np.sum(AAs[:,2])/temp))
    print("OA值为: \n{:.3f}".format(np.sum(OAs)/temp))
    print("kappa值为: \n{:.3f}".format(np.sum(kappas)/temp))
```

2. sonar数据集分类（数据来源见附件sonar.xlsx）

```
# -*- coding: utf-8 -*-

from sklearn.model_selection import train_test_split
import numpy as np
from matplotlib import pyplot as plt
import random
from openpyxl import load_workbook


# 计算并返回均值向量
def junzhi(sonar):
    a = np.zeros([60, 1])
```

```
    for i in range(60):
        a[i]=np.mean(sonar[:,i])
    return a

# 计算类内离散度矩阵S_i
def S_i(sonar):
    a = junzhi(sonar)
    b = np.zeros([sonar.shape[1], sonar.shape[1]])
    for i in range(sonar.shape[0]):
        b = b + np.matmul((sonar[i, :].T - a), (sonar[i, :].T - a).T)
    return b

# 计算类间离散度矩阵S_b
def S_b(sonar1, sonar2):
    b_1 = S_i(sonar1)
    b_2 = S_i(sonar2)
    c = b_1 + b_2
    return c

# 划分训练集、测试集的函数
def train_test(sonar, target, num):
    train_sonar, test_sonar, train_target, test_target =\
        train_test_split(sonar, target, test_size=0.6, random_state=num,
shuffle=True)
    return {'train_sonar': train_sonar, 'test_sonar': test_sonar,
'train_target': train_target, 'test_target': test_target}

# 计算出最优的投影方向并计算出决策点
def best_w(sonar1, sonar2, target1, target2, num):
    train_sonar1 = train_test(sonar1, target1, num)['train_sonar']
    train_sonar2 = train_test(sonar2, target2, num)['train_sonar']
    s_0 = S_b(train_sonar1, train_sonar2)
    best_w = np.matmul(np.linalg.inv(s_0), junzhi(train_sonar1) -
junzhi(train_sonar2))
    y_0 = (58/124)*np.mean(np.matmul(train_sonar1, best_w)) +
(66/124)*np.mean(np.matmul(train_sonar2, best_w))
    # print(best_w)
    return best_w, y_0

# 对测试样本进行分类并计算相关评价指标
def classify(sonar1,sonar2,target1,target2,num):
    # print(num)
    ## 训练集得到的最佳方向和决策点
    w_best12,y0_12=best_w(sonar1, sonar2, target1, target2, num)
    ## 测试集
    test1=train_test(sonar1,target1,num)['test_sonar']
    test2=train_test(sonar2,target2,num)['test_sonar']
    test=np.vstack((test1,test2))
    ## 当前测试集对应的标签
    test_target1=train_test(sonar1,target1,num)['test_target']
    test_target2=train_test(sonar2,target2,num)['test_target']
    # print(test_target1.shape)
    # print(test_target2.shape)
    test_target=np.vstack((test_target1,test_target2))
    # print(test_target.shape)
    # print(test_target)
    ## 存放预测得到的标签
    predict_target=np.zeros_like(test_target)
```

```
    ## 通过决策函数
    y=np.matmul(test,w_best12)
    for i in range(len(test)):
        if y[i]>y0_12 or y[i]==y0_12:
            predict_target[i]=0
        else:
            predict_target[i]=1
    # print(predict_target)
    ## 计算OA、AA、kappa系数
    ### 记录三类样本分类正确的数量
    num_1=0
    num_2=0
    ### 记录三类样本实际分类的数量
    real_num_1=0
    real_num_2=0
    for i in range(len(test_target)):
        ## 统计分类正确的数量
        if i<len(test_target1):
            if predict_target[i]==test_target[i]:
                num_1=num_1+1
        else:
            if predict_target[i]==test_target[i]:
                num_2=num_2+1
        ## 统计实际的分类数量
        if predict_target[i]==0:
            real_num_1=real_num_1+1
        else:
            real_num_2=real_num_2+1
    # print(num_1)
    ### 计算相关指标
    OA=(num_1+num_2)/len(test_target)
    AA_1=(num_1)/len(test_target1)
    AA_2=(num_2)/len(test_target2)
    pe=(real_num_1*len(test_target1)+real_num_2*\
        len(test_target2))/np.square(len(test_target))
    kappa=(OA-pe)/(1-pe)
    return OA,[AA_1,AA_2],kappa


if __name__ == "__main__":
    # 导入数据
    workbook=load_workbook(filename='sonar.xlsx')
    # print(workbook.sheetnames)
    sheet=workbook['Sheet1']
    # 存储样本特征集
    sonar=np.zeros([208,60])
    # 存储标签
    target=np.zeros([208,1])
    for i in range(208):
        for j in range(60):
            sonar[i,j]=sheet.cell(row=i+1,column=j+1).value
    # print(sonar.shape)
    # print(sonar[0,59])
    for i in range(208):
        if sheet.cell(row=i+1,column=61).value=='R':
            target[i]=0
        else:
            target[i]=1
```

```
    # print(target[97])
    sonar1=sonar[0:97,:]
    sonar2=sonar[97:208,:]
    target1=target[0:97,:]
    target2=target[97:208,:]
    # 存储相关的指标值
    OAs=np.zeros([30,1])
    AAs=np.zeros([30,2])
    kappas=np.zeros([30,1])
    #随机给出30个随机种子，用于train_test函数
    # nums=
[703,5205,8248,4998,1027,8528,7063,6513,793,2805,1524,8985,3939,9000\
    #
,3796,3178,628,9359,582,265,5920,8866,7960,5090,5481,4928,526,8763,5333,659
6]
    nums=random.sample(range(0,10000),100)
    j=0;k=0
    while j<30:
        try:

 OAs[j],AAs[j,],kappas[j]=classify(sonar1,sonar2,target1,target2,nums[k])
            j=j+1
            k=k+1
        except:
            k+=1

    #
OAs[j],AAs[j,],kappas[j]=classify(sonar1,sonar2,target1,target2,nums[k])

    temp=0
    for i in range(len(OAs)):
        if OAs[i]!=0:
            temp=temp+1
    print(OAs)
    print(AAs)
    print(kappas)
    print("AA值分别为：
\n{:.3f}\n{:.3f}".format(np.sum(AAs[:,0])/temp,np.sum(AAs[:,1]/temp)))
    print("OA值为：\n{:.3f}".format(np.sum(OAs)/temp))
    print("kappa值为：\n{:.3f}".format(np.sum(kappas)/temp))
```