

实验一：Fisher 辨别分析

1 问题描述

使用 Fisher 线性判别处理分类问题，并在 UCI 数据集上的 Iris 和 sonar 数据上验证算法的有效性，其中，Iris 数据 3 类，4 维，150 个数据；sonar 数据 2 类，60 维，208 个样本。

2 数据集说明

UCI 数据库是加州大学欧文分校 (University of CaliforniaIrvine) 提出的用于机器学习的数据库，这个数据库目前共有 488 个数据集，其数目还在不断增加，UCI 数据集是一个常用的标准测试数据集。

下表为部分 Iris 数据展示：

5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
...				
7	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
...				
6.3	3.3	6	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3	5.9	2.1	virginica

3 Fisher 原理分析

判别分析是一种经典的现行分析方法，其利用已知类别的样本建立判别模型，对未知类别的样本进行分类。

3.1 Fisher 思想

Fisher 的主要思想是投影，选择一个适当的投影轴，使所有的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来处理。在计算投影轴的方向时，需保证每一类内的投影距离尽可能小，而不同类的投影距离尽可能大。

3.2 公式推导

Fisher 线性判别需根据实际数据找到一条最易于分类的投影方向。

- 在 n 维 X 空间

- 各类样本的均值向量 $\mu_i : \mu_i = \frac{1}{N_i} \sum_{\mathbf{x}_j \in D_i} \mathbf{x}_j, i = 1, 2$
- 样本类内离散度矩阵 S_i 和总样本类内离散度矩阵 S_w

$$S_i = \sum_{\mathbf{x}_j \in D_i} (\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T, i = 1, 2$$

$$S_w = S_1 + S_2$$

3. 样本类间离散度矩阵 S_b

$$S_b = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

• 在一维 Y 空间

1. 各类样本的均值向量 $\bar{\mu}_i: \bar{\mu}_i = \frac{1}{N_i} \sum_{y_j \in D'_i} y_j, i = 1, 2$

2. 样本类内离散度矩阵 \bar{S}_i 和总样本类内离散度矩阵 \bar{S}_w

$$\bar{S}_i^2 = \sum_{y_j \in D'_i} (y_j - \bar{\mu}_i)^2, i = 1, 2$$

$$\bar{S}_w = \bar{S}_1^2 + \bar{S}_2^2$$

3. 样本类间离散度矩阵 \bar{S}_b

$$\bar{S}_b = (\bar{\mu}_1 - \bar{\mu}_2)^2$$

目标：投影后，在一维 Y 空间中各类样本尽可能分得开些，即使原样本向量在该方向上的投影能兼顾类间分布尽可能分开，类内样本投影尽可能密集的要求。

由此构造 Fisher 准则函数为

$$J(\mathbf{w}) = \frac{\bar{S}_b}{\bar{S}_w} = \frac{(\bar{\mu}_1 - \bar{\mu}_2)^2}{\bar{S}_1^2 + \bar{S}_2^2}$$

Fisher 最佳投影方向的求解为

$$\mathbf{w}^* = \operatorname{argmax} J(\mathbf{w})$$

将 $J(\mathbf{w})$ 变成 \mathbf{w} 的显函数之后，

由各类样本均值可推出：

$$\bar{\mu}_i = \frac{1}{N_i} \sum_{y_j \in D'_i} y_j = \frac{1}{N_i} \sum_{\mathbf{x}_j \in D_i} \mathbf{w}^T \mathbf{x}_j = \mathbf{w}^T \left(\frac{1}{N_i} \sum_{\mathbf{x}_j \in D_i} \mathbf{x}_j \right) = \mathbf{w}^T \mu_i$$

投影样本均值之差可以展开为：

$$\begin{aligned} (\bar{\mu}_1 - \bar{\mu}_2)^2 &= (\mathbf{w}^T \mu_1 - \mathbf{w}^T \mu_2)^2 \\ &= \mathbf{w}^T (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \mathbf{w} \\ &= \mathbf{w}^T \mathbf{S}_b \mathbf{w} \end{aligned}$$

由类内散布矩阵可推出：

$$\begin{aligned} \bar{S}_i^2 &= \sum_{y_j \in D'_i} (y_j - \bar{\mu}_i)^2 = \sum_{\mathbf{x}_j \in D_i} (\mathbf{w}^T \mathbf{x}_j - \mathbf{w}^T \mu_i)^2 \\ &= \mathbf{w}^T \left[\sum_{\mathbf{x}_j \in D_i} (\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T \right] \mathbf{w} \\ &= \mathbf{w}^T \mathbf{S}_i \mathbf{w} \end{aligned}$$

于是有：

$$\bar{S}_1^2 + \bar{S}_2^2 = \mathbf{w}^T (\mathbf{S}_1 + \mathbf{S}_2) \mathbf{w} = \mathbf{w}^T \mathbf{S}_w \mathbf{w}$$

准则函数可以写为：

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_b \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}}$$

要求使 $J(\mathbf{w})$ 最大的 \mathbf{w} ，可以采用 Lagrange 乘子法求解。

假设分母等于非零常数，定义 Lagrange 函数为：

$$L(\mathbf{w}, \lambda) = \mathbf{w}^T \mathbf{S}_b \mathbf{w} - \lambda (\mathbf{w}^T \mathbf{S}_w \mathbf{w} - c)$$

对 \mathbf{w} 求偏导数，令偏导数为 0：

$$\frac{\partial L(\mathbf{w}, \lambda)}{\partial \mathbf{w}} = 2\mathbf{S}_b \mathbf{w} - 2\lambda \mathbf{S}_w \mathbf{w} = 0$$

即：

$$\mathbf{S}_w^{-1} \mathbf{S}_b \mathbf{w}^* = \lambda \mathbf{w}^*$$

化简得：

$$\mathbf{w}^* = \frac{1}{\lambda} \mathbf{S}_w^{-1} \mathbf{S}_b \mathbf{w}^* = \frac{1}{\lambda} \mathbf{S}_w^{-1} (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \mathbf{w}^*$$

该式中， $(\mu_1 - \mu_2)^T \mathbf{w}^*$ 和 λ 为标量，对于投影向量，我们仅需考虑它的方向，因此

$$\mathbf{w}^* = \mathbf{S}_w^{-1} (\mu_1 - \mu_2)$$

最后通过 $y = w^{*T} x$ 对样本集作线性变换得到 n 个样本投影后的样本值 y_1, y_2, \dots, y_n ，对这些投影之后的样本值进行分类，确定一个阈值 w_0 ，分类规则如下：

$$y > w_0 \rightarrow x \in \omega_1$$

$$y < w_0 \rightarrow x \in \omega_2$$

阈值可以通过下面三种方法进行取值：

$$w_0 = \frac{\bar{\mu}_1 + \bar{\mu}_2}{2}$$

$$w_0 = \frac{N_1 \bar{\mu}_1 + N_2 \bar{\mu}_2}{N_1 + N_2}$$

$$w_0 = \frac{\bar{\mu}_1 + \bar{\mu}_2}{2} + \frac{\ln [P(\omega_1) / P(\omega_2)]}{N_1 + N_2 - 2}$$

本次实验阈值根据第一种方式进行选取。

4 实验结果

4.1 Iris 数据集（3 类，4 维）

本次实验对 Iris 数据集采用了三种方法求解分类准确率，结果如下表所示：

表 1: Iris 数据集三种分类方法的准确率

分类方法	准确率
留出法	97.11%
10 折交叉验证法	97.33%
留一法	98.00%

其中，留出法对训练集和测试集按照 7/3 的比例进行随机划分，计算 10 次准确率之后求取平均值。

由于 fish 线性判别单次运行只能投影两类样本，因此，建立了三个最佳投影向量，使数据能够进行两两投影比较。Iris 数据集分别在三个投影向量上的投影结果如下图所示。

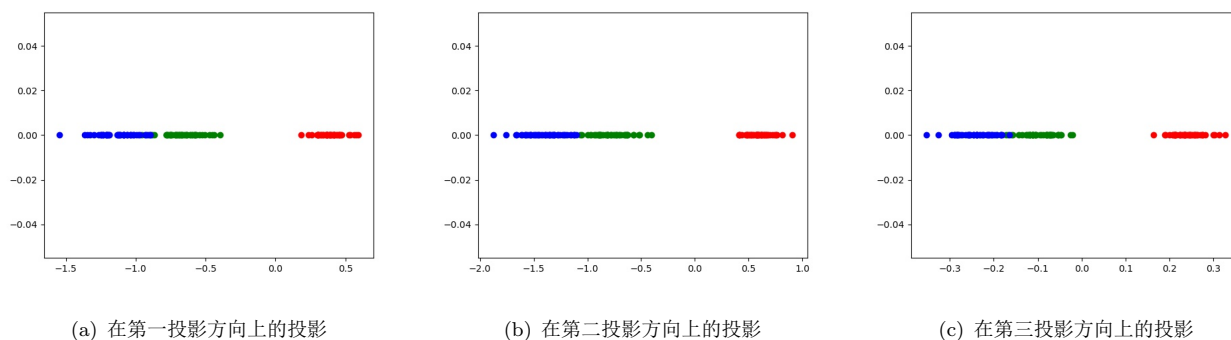


图 1: Iris 数据集在三类投影方向上的投影

从图中可以看出, Iris 数据集在三个方向上投影分隔比较明显, 重叠部分很少, 这印证了该分类算法在 Iris 数据集上准确率很高。

4.2 sonar 数据集 (2 类, 60 维)

本次实验对 sonar 数据集采用了三种方法求解分类准确率, 结果如下表所示:

表 2: sonar 数据集三种分类方法的准确率

分类方法	准确率
留出法	72.22%
10 折交叉验证法	59.19%
留一法	75.00%

其中, 留出法对训练集和测试集按照 7/3 的比例进行随机划分, 计算 10 次准确率之后求取平均值。

从上表中可以发现, 对于 K 折交叉验证法, 取 K 为 10 时, 准确率明显较低。当增大 K 时, 准确率得到明显提高, 当 K 取值为 80 时, K 折交叉验证法准确率逼近留一法。考虑到留一法是特殊的 K 折交叉验证法 (即 $K=N-1$), 说明当样本量较大时, 适当增大 K 值有利于提升该分类算法的准确率。

sonar 数据集在投影向量上的投影结果如下图所示。

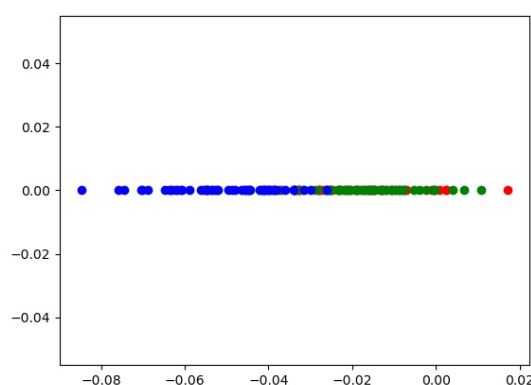


图 2: sonar 数据集在三类投影方向上的投影

从图中可以看出, 三类样本无明显间隔区分, 部分混叠严重, 这也印证了该分类算法在 sonar 数据集上准确率不高。

4.3 分类准确率和维度之间的关系

为了进一步探究分类准确率和维度之间的关系，本实验选取了 60 维的 sonar 数据集作为研究对象，采取了分类准确率最高的留一法，得到结果如下图所示：

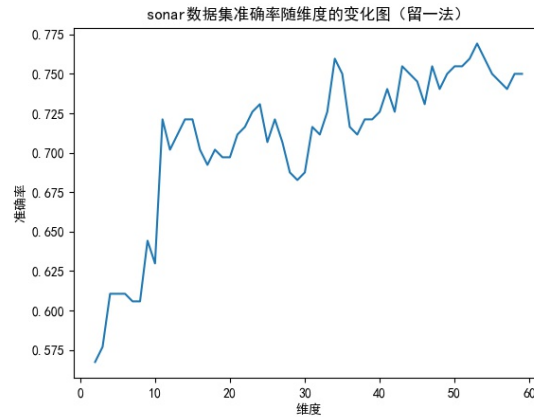


图 3: sonar 数据集上分类准确率和维度之间的关系

从图中可以得出结论，随着维度的升高，该算法的分类准确率也随之增大，但分类准确率的增大幅度逐渐减缓，并且，增大幅度具有波动性。

维度的升高，会使算法的计算成本增大，因此，可以在图中选取一个拐点作为最佳维度的选取点。对于 sonar 数据集来说，当维度为 11 时，准确率已经达到了 72.12%，在这一点之后，准确率的增大幅度趋于缓慢。因此，在本实验中，维度最适合的取值为 11。

5 程序代码

对 Iris 数据集分类的代码如下：

```
1 import numpy as np
2 from sklearn.model_selection import train_test_split, KFold, LeaveOneOut
3 import matplotlib.pyplot as plt
4
5
6 def load_dataset():
7     data = np.genfromtxt('iris.txt', delimiter=',', usecols=(0, 1, 2, 3))
8     target = np.genfromtxt('iris.txt', delimiter=',', usecols=(4), dtype=str)
9     t = np.zeros(len(target))
10    t[target == 'setosa'] = 1
11    t[target == 'versicolor'] = 2
12    t[target == 'virginica'] = 3
13    return data, t
14
15
16 def fisher(class1, class2):
17     class1 = np.mat(class1)
18     class2 = np.mat(class2)
```

```

19
20 # 求解每一个特征的均值，按列求解
21 a1 = np.mean(class1, axis=0)
22 a2 = np.mean(class2, axis=0)
23
24 # 直接代入公式求解类内离散度矩阵
25 s1 = (class1 - a1).T * (class1 - a1)
26 s2 = (class2 - a2).T * (class2 - a1)
27 sw = s1 + s2
28 # 这里是求解离散度矩阵的另一种思路：通过协方差公式求解，49为样本数量-1(n
    -1)
29 # s = np.cov(class0.T) * 49
30
31 # w 为最佳变换向量w*, w0为阈值
32 w = (a1 - a2) * np.linalg.inv(sw)
33 w0 = (a1 * w.T + a2 * w.T) / 2
34 return w, w0
35
36
37 # 计算分类准确率
38 def accuracy(pre, tar):
39     total = len(pre)
40     acc = 0
41     for i in range(total):
42         if pre[i] == tar[i]:
43             acc += 1
44     return acc / total
45
46
47 # 修改三个类别标签
48 def transform_target(data, target):
49     class1 = []
50     class2 = []
51     class3 = []
52     for i in range(len(data)):
53         if target[i] == 1:
54             class1.append(data[i])
55         elif target[i] == 2:
56             class2.append(data[i])
57         else:
58             class3.append(data[i])
59     return class1, class2, class3
60
61

```

```

62 # method1 留出法，随机划分训练测试集，多次平均求结果
63 def method1():
64     data, target = load_dataset()
65
66     # 使用留出法随机划分数据集，训练集/测试集=7/3，每次划分具有随机性
67     X_train, X_test, Y_train, Y_test = train_test_split(data, target,
68                                                         test_size=0.30)
69
70     class1, class2, class3 = transform_target(X_train, Y_train)
71     # w12代表第一类和第二类比较的投影向量，w012代表第一类和第二类比较时的阈
72     # 值，其它同理。
73     w12, w012 = fisher(class1, class2)
74     w13, w013 = fisher(class1, class3)
75     w23, w023 = fisher(class2, class3)
76
77     # 3分类的比较思路：两两进行比较，若两次均分类正确才算正确
78     y12 = X_test * w12.T
79     y13 = X_test * w13.T
80     y23 = X_test * w23.T
81     res = np.zeros(len(X_test))
82     for i in range(len(res)):
83         if y12[i] > w012 and y12[i] > w013:
84             res[i] = 1
85         if y12[i] < w012 and y23[i] > w023:
86             res[i] = 2
87         if y13[i] < w013 and y23[i] < w023:
88             res[i] = 3
89     # print(res)
90     acc = accuracy(res, Y_test)
91     # print("分类准确率为", acc)
92     return acc
93
94 # method2 k折交叉验证法
95 def method2():
96     data, target = load_dataset()
97     acc = 0
98     K = 10 # 这里设定k为10
99     kf = KFold(n_splits=K)
100     for train_index, test_index in kf.split(data):
101         X_train = data[train_index]
102         X_test = data[test_index]
103         Y_train = target[train_index]
104         Y_test = target[test_index]

```

```

104 class1, class2, class3 = transform_target(X_train, Y_train)
105 # w12代表第一类和第二类比较的投影向量，w012代表第一类和第二类比较时的阈
    值，其它同理。
106 w12, w012 = fisher(class1, class2)
107 w13, w013 = fisher(class1, class3)
108 w23, w023 = fisher(class2, class3)
109
110 # 3分类的比较思路：两两进行比较，若两次均分类正确才算正确
111 y12 = X_test * w12.T
112 y13 = X_test * w13.T
113 y23 = X_test * w23.T
114 res = np.zeros(len(X_test))
115 for i in range(len(res)):
116     if y12[i] > w012 and y12[i] > w013:
117         res[i] = 1
118     if y12[i] < w012 and y23[i] > w023:
119         res[i] = 2
120     if y13[i] < w013 and y23[i] < w023:
121         res[i] = 3
122     # print(res)
123     acc += accuracy(res, Y_test)
124 # print("分类准确率为", acc)
125 acc = acc / K
126 return acc
127
128 # method3 留一法
129 def method3():
130     data, target = load_dataset()
131     loo = LeaveOneOut()
132     acc = 0
133     for train_index, test_index in loo.split(data):
134         X_train = data[train_index]
135         X_test = data[test_index]
136         Y_train = target[train_index]
137         Y_test = target[test_index]
138         class1, class2, class3 = transform_target(X_train, Y_train)
139         # w12代表第一类和第二类比较的投影向量，w012代表第一类和第二类比较时的阈
            值，其它同理。
140         w12, w012 = fisher(class1, class2)
141         w13, w013 = fisher(class1, class3)
142         w23, w023 = fisher(class2, class3)
143
144         # 3分类的比较思路：两两进行比较，若两次均分类正确才算正确
145         y12 = X_test * w12.T

```



```

146 y13 = X_test * w13.T
147 y23 = X_test * w23.T
148 res = np.zeros(len(X_test))
149 for i in range(len(res)):
150     if y12[i] > w012 and y12[i] > w013:
151         res[i] = 1
152     if y12[i] < w012 and y23[i] > w023:
153         res[i] = 2
154     if y13[i] < w013 and y23[i] < w023:
155         res[i] = 3
156     # print(res)
157     acc += accuracy(res, Y_test)
158     # print("分类准确率为", acc)
159     acc = acc / len(data)
160     return acc
161
162 # 绘制投影图
163 def draw():
164     data, target = load_dataset()
165     class1, class2, class3 = transform_target(data, target)
166     # w12代表第一类和第二类比较的投影向量，w012代表第一类和第二类比较时的阈
        值，其它同理。
167     w12, w012 = fisher(class1, class2)
168     w13, w013 = fisher(class1, class3)
169     w23, w023 = fisher(class2, class3)
170
171     # 3分类的比较思路：两两进行比较，若两次均分类正确才算正确
172     y12 = data * w12.T
173     y13 = data * w13.T
174     y23 = data * w23.T
175
176     # y12方向上的投影
177     plt.figure(1)
178     plt.plot(y12[0:49], np.zeros([49, 1]), 'ro')
179     plt.plot(y12[50:99], np.zeros([49, 1]), 'go')
180     plt.plot(y12[100:149], np.zeros([49, 1]), 'bo')
181     plt.savefig('./iris-1.jpg')
182     plt.show()
183
184
185     # y13方向上的投影
186     plt.figure(2)
187     plt.plot(y13[0:49], np.zeros([49, 1]), 'ro')
188     plt.plot(y13[50:99], np.zeros([49, 1]), 'go')

```

```

189 plt.plot(y13[100:149], np.zeros([49, 1]), 'bo')
190 plt.savefig('./iris-2.jpg')
191 plt.show()
192
193
194 # y23方向上的投影
195 plt.figure(3)
196 plt.plot(y23[0:49], np.zeros([49, 1]), 'ro')
197 plt.plot(y23[50:99], np.zeros([49, 1]), 'go')
198 plt.plot(y23[100:149], np.zeros([49, 1]), 'bo')
199 plt.savefig('./iris-3.jpg')
200 plt.show()
201
202
203
204 def main():
205     # 10次计算方法一的留出法，取平均准确率作为结果(保留两位小数输出)
206     total_accuary = 0
207     for i in range(10):
208         total_accuary += method1()
209     total_accuary = total_accuary / 10
210     print("留出法的分类准确率为：", "{:.2%}".format(total_accuary))
211     draw()
212     total_accuary2 = method2()
213     print("10折交叉验证法的分类准确率为：", "{:.2%}".format(total_accuary2))
214     total_accuary3 = method3()
215     print("留一法的分类准确率为：", "{:.2%}".format(total_accuary3))
216
217 if __name__ == '__main__':
218     main()

```

对 sonar 数据集分类的代码如下：

```

1 import numpy as np
2 from sklearn.model_selection import train_test_split, KFold, LeaveOneOut
3 import matplotlib.pyplot as plt
4
5
6 # 正常导入数据
7 def load_dataset():
8     data = np.genfromtxt('sonar.txt', delimiter=',', usecols=np.arange(0, 59)
9         )
9     target = np.genfromtxt('sonar.txt', delimiter=',', usecols=(60), dtype=
10         str)
10 t = np.zeros(len(target))

```

```

11 t[target == 'R'] = 1
12 t[target == 'M'] = 2
13 return data, t
14
15
16 # 自定义导入数据维度
17 def load_dataset_dimension(dimension):
18     data = np.genfromtxt('sonar.txt', delimiter=',', usecols=np.arange(0,
19                             dimension))
19     target = np.genfromtxt('sonar.txt', delimiter=',', usecols=(60), dtype=
20                             str)
21     t = np.zeros(len(target))
22     t[target == 'R'] = 1
23     t[target == 'M'] = 2
24     return data, t
25
26 def fisher(class1, class2):
27     class1 = np.mat(class1)
28     class2 = np.mat(class2)
29
30     # 求解每一个特征的均值，按列求解
31     a1 = np.mean(class1, axis=0)
32     a2 = np.mean(class2, axis=0)
33
34     # 直接代入公式求解类内离散度矩阵
35     s1 = (class1 - a1).T * (class1 - a1)
36     s2 = (class2 - a2).T * (class2 - a1)
37     sw = s1 + s2
38     # 这里是求解离散度矩阵的另一种思路：通过协方差公式求解，49为样本数量-1(n
39         -1)
40     # s = np.cov(class0.T) * 49
41
42     # w 为最佳变换向量w*, w0为阈值
43     w = (a1 - a2) * np.linalg.inv(sw)
44     w0 = (a1 * w.T + a2 * w.T) / 2
45     return w, w0
46
47 # 计算分类准确率
48 def accuracy(pre, tar):
49     total = len(pre)
50     acc = 0
51     for i in range(total):

```

```

52  if pre[i] == tar[i]:
53      acc += 1
54  return acc / total
55
56
57  # 修改两个类别标签
58  def transform_target(data, target):
59      class1 = []
60      class2 = []
61      for i in range(len(data)):
62          if target[i] == 1:
63              class1.append(data[i])
64          elif target[i] == 2:
65              class2.append(data[i])
66      return class1, class2
67
68
69  # method1 留出法，随机划分训练测试集，多次平均求结果
70  def method1():
71      data, target = load_dataset()
72
73      # 使用留出法随机划分数据集，训练集/测试集=7/3，每次划分具有随机性
74      X_train, X_test, Y_train, Y_test = train_test_split(data, target,
75                                                             test_size=0.30)
76
77      class1, class2 = transform_target(X_train, Y_train)
78      # w代表投影向量，w0代表第一类和第二类比较时的阈值。
79      w, w0 = fisher(class1, class2)
80
81      y = X_test * w.T
82      res = np.zeros(len(X_test))
83      for i in range(len(res)):
84          if y[i] > w0:
85              res[i] = 1
86          else:
87              res[i] = 2
88      # print(res)
89      acc = accuracy(res, Y_test)
90      # print("分类准确率为", acc)
91      return acc
92
93  # method2 k折交叉验证法
94  def method2():

```

```

95 data, target = load_dataset()
96 acc = 0
97 K = 10 # 这里设定k为10
98 kf = KFold(n_splits=K)
99 for train_index, test_index in kf.split(data):
100 X_train = data[train_index]
101 X_test = data[test_index]
102 Y_train = target[train_index]
103 Y_test = target[test_index]
104 class1, class2 = transform_target(X_train, Y_train)
105 # w代表投影向量, w0代表第一类和第二类比较时的阈值。
106 w, w0 = fisher(class1, class2)
107
108 y = X_test * w.T
109 res = np.zeros(len(X_test))
110 for i in range(len(res)):
111     if y[i] > w0:
112         res[i] = 1
113     else:
114         res[i] = 2
115 # print(res)
116 acc += accuracy(res, Y_test)
117 # print("分类准确率为", acc)
118 acc = acc / K
119 return acc
120
121
122 # method3 留一法
123 def method3():
124     data, target = load_dataset()
125     loo = LeaveOneOut()
126     acc = 0
127     for train_index, test_index in loo.split(data):
128         X_train = data[train_index]
129         X_test = data[test_index]
130         Y_train = target[train_index]
131         Y_test = target[test_index]
132         class1, class2 = transform_target(X_train, Y_train)
133         # w代表投影向量, w0代表第一类和第二类比较时的阈值。
134         w, w0 = fisher(class1, class2)
135
136         y = X_test * w.T
137         res = np.zeros(len(X_test))
138         for i in range(len(res)):

```

```

139 if y[i] > w0:
140     res[i] = 1
141 else:
142     res[i] = 2
143 # print(res)
144 acc += accuracy(res, Y_test)
145 # print("分类准确率为", acc)
146 acc = acc / len(data)
147 return acc
148
149
150 # testdension 以留一法为基础，测试维度和准确率的关系
151 def testdension(dimension):
152     data, target = load_dataset_dimension(dimension)
153     loo = LeaveOneOut()
154     acc = 0
155     for train_index, test_index in loo.split(data):
156         X_train = data[train_index]
157         X_test = data[test_index]
158         Y_train = target[train_index]
159         Y_test = target[test_index]
160         class1, class2 = transform_target(X_train, Y_train)
161         # w代表投影向量，w0代表第一类和第二类比较时的阈值。
162         w, w0 = fisher(class1, class2)
163
164         y = X_test * w.T
165         res = np.zeros(len(X_test))
166         for i in range(len(res)):
167             if y[i] > w0:
168                 res[i] = 1
169             else:
170                 res[i] = 2
171         # print(res)
172         acc += accuracy(res, Y_test)
173         # print("分类准确率为", acc)
174         acc = acc / len(data)
175     return acc
176
177
178 # 绘制投影图
179 def draw():
180     data, target = load_dataset()
181     class1, class2 = transform_target(data, target)
182

```

```

183 w, w0 = fisher(class1, class2)
184 y = data * w.T
185
186 plt.figure(1)
187 plt.plot(y[0:49], np.zeros([49, 1]), 'ro')
188 plt.plot(y[50:99], np.zeros([49, 1]), 'go')
189 plt.plot(y[100:149], np.zeros([49, 1]), 'bo')
190 plt.savefig('./sonar.jpg')
191 plt.show()
192
193
194 def main():
195     # 10次计算方法一的留出法，取平均准确率作为结果(保留两位小数输出)
196     total_accuary1 = 0
197     for i in range(10):
198         total_accuary1 += method1()
199     total_accuary1 = total_accuary1 / 10
200     print("留出法的分类准确率为：", "{:.2%}".format(total_accuary1))
201     # draw()
202     total_accuary2 = method2()
203     print("K折交叉验证法的分类准确率为：", "{:.2%}".format(total_accuary2))
204     total_accuary3 = method3()
205     print("留一法的分类准确率为：", "{:.2%}".format(total_accuary3))
206
207
208 # 绘制维度与准确率的关系图
209 total_accuary = []
210 plt.figure(2)
211 for demension in range(2, 60):
212     total_accuary.append(testdension(demension))
213     print(total_accuary)
214     plt.plot(np.arange(2, 60), total_accuary)
215 # 解决中文显示问题
216 plt.rcParams['font.sans-serif'] = ['SimHei']
217 plt.rcParams['axes.unicode_minus'] = False
218 plt.xlabel("维度")
219 plt.ylabel("准确率")
220 plt.title("sonar数据集准确率随维度的变化图（留一法）")
221 plt.savefig('./demension.jpg')
222 plt.show()
223
224
225
226 if __name__ == '__main__':

```

