

模式识别课后作业三

2020E8013282019 李一帆

November 5, 2020

1 计算与证明

1.1 问题 1

将四个训练样本表示为规范化增广样本有： $(1, 4, 1)^T, (2, 3, 1)^T, (-4, -1, -1)^T, (-3, -2, -1)^T$ 。分别记作 x_1, x_2, x_3, x_4 。根据批处理感知器算法，由于 $a^T \cdot x_1 > 0, a^T \cdot x_2 > 0, a^T \cdot x_3 < 0, a^T \cdot x_4 < 0$ ，因此第一次迭代可以表示为：

$$\begin{aligned} a &:= a + 1 \cdot \sum_{i=3}^4 x_i \\ &= \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} -4 \\ -1 \\ -1 \end{pmatrix} + \begin{pmatrix} -3 \\ -2 \\ -1 \end{pmatrix} \\ &= \begin{pmatrix} -7 \\ -2 \\ -2 \end{pmatrix} \end{aligned}$$

同理，由于 $a^T \cdot x_1 < 0, a^T \cdot x_2 < 0, a^T \cdot x_3 > 0, a^T \cdot x_4 > 0$ ，因此第二次迭代可以表示为如下形式。

$$\begin{aligned} a &:= a + 1 \cdot \sum_{i=1}^2 x_i \\ &= \begin{pmatrix} -7 \\ -2 \\ -2 \end{pmatrix} + \begin{pmatrix} 1 \\ 4 \\ 1 \end{pmatrix} + \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} -4 \\ 5 \\ 0 \end{pmatrix} \end{aligned}$$

因为此时 $a^T \cdot x_1 > 0, a^T \cdot x_2 > 0, a^T \cdot x_3 > 0, a^T \cdot x_4 > 0$ ，满足递归结束条件。因此，最终得到的权向量 $a = (-4, 5, 0)^T$ 。

1.2 问题 2

根据决策规则如果 $g_i(\mathbf{x}) > g_j(\mathbf{x})$, 则 \mathbf{x} 被划分为 ω_i 类别。

因此当 $\begin{cases} g_1(x) > g_2(x) \\ g_1(x) > g_3(x) \end{cases} \Rightarrow \begin{cases} x_1 < \frac{1}{2} \\ x_2 > \frac{x_1}{2} \end{cases}$ 时, 属于 ω_1 ;

当 $\begin{cases} g_2(x) > g_1(x) \\ g_2(x) > g_3(x) \end{cases} \Rightarrow \begin{cases} x_1 > \frac{1}{2} \\ x_2 > \frac{1-x_1}{2} \end{cases}$ 时, 属于 ω_2 ;

当 $\begin{cases} g_3(x) > g_1(x) \\ g_3(x) > g_2(x) \end{cases} \Rightarrow \begin{cases} x_2 < \frac{x_1}{2} \\ x_2 < \frac{1-x_1}{2} \end{cases}$ 时, 属于 ω_3 。

根据上述关系可以画出决策面如图 Figure 1 所示。

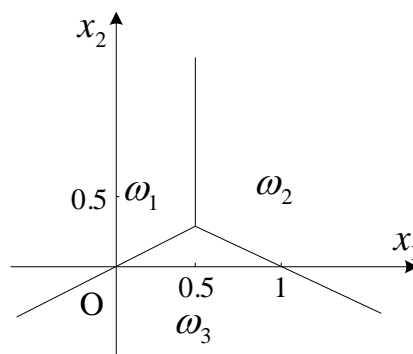


Figure 1: 决策面.

由 Figure 1 可以看出, 由于所有区域都被划分成了三类中的某一类, 不存在未被划分的区域, 也不存在二义性区域。因此, 此时不存在分类不确定区域。

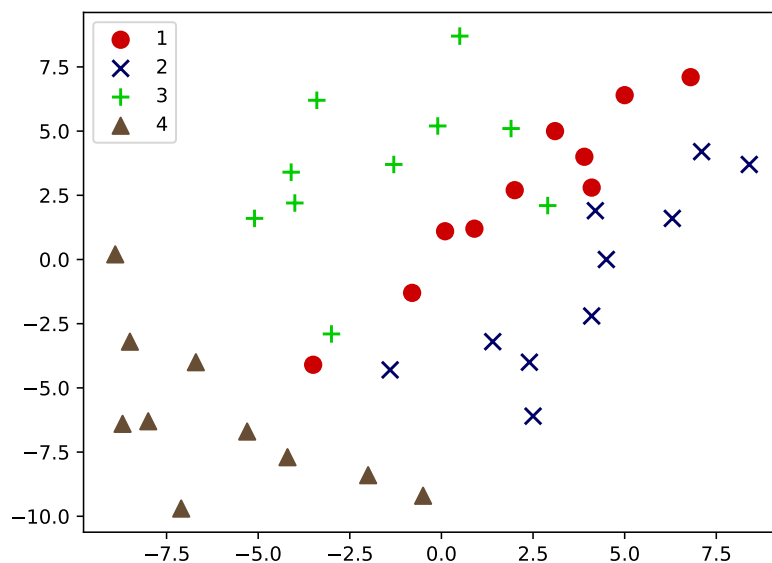


Figure 2: 数据可视化.

2 计算机编程

将数据进行可视化，如 Figure 2 所示。

2.1 问题 1

Batch Perception 的算法和 Python 代码如下所示。

Algorithm 1: Batch Perceptron

```
1 begin initialize:  $a, \eta$ , certain  $\theta$ (small value),  $k = 0$ ;  
2 while  $|\eta_k \sum y| < \theta, y \in Y_k$  do  
3    $k \leftarrow k + 1$ ;  
4    $a = a + \eta_k \sum_{y \in Y(k)} y$ ;  
5   return  $a$ ;
```

```
1 def batch_perceptron(lr=0.5, data=None):  
2     a = np.zeros([3, 1])  
3     threshold = 1e-9  
4     data[:, 2] = 1  
5     data[10:, :] = -data[10:, :]  
6     iter = 0  
7     for i in range(1000000):  
8         result = a.T.dot(data.T)  
9         _, inx = np.where(result <= 0)  
10        increments = np.sum(lr * data[inx], axis=0)  
11        a += increments.reshape([3, 1])  
12        iter += 1  
13        if abs(increments.sum()) < threshold:  
14            print('total iter:%d' % iter)  
15            return a
```

2.1.1 问题 a

对 ω_1 和 ω_2 的分类结果如 Figure 3 所示。

通过运行上述算法，可以得到对 ω_1 和 ω_2 的迭代次数为 24。

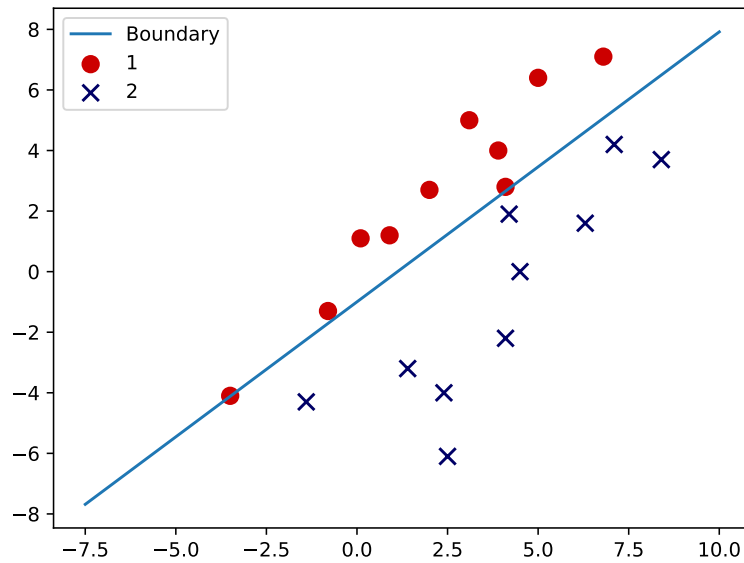


Figure 3: Batch perceptron 分类 ω_1 和 ω_2 .

2.1.2 问题 b

对 ω_2 和 ω_3 的分类结果如 Figure 4 所示，迭代次数为 17 次。

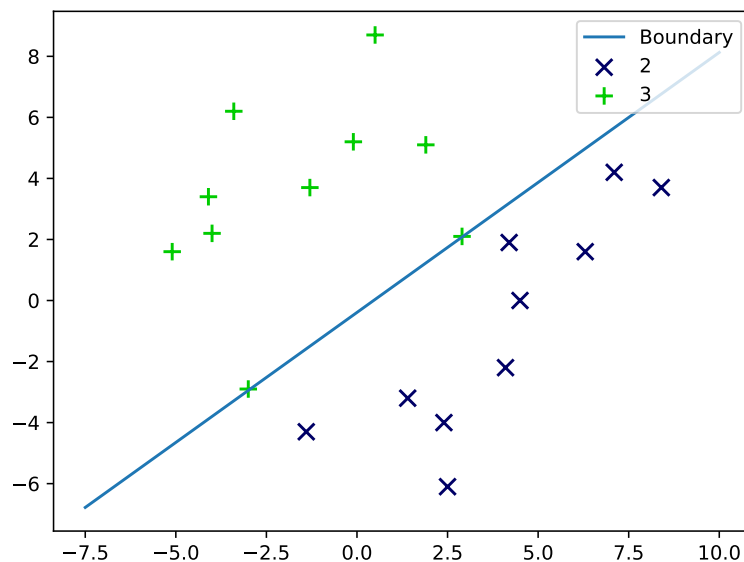


Figure 4: Batch perceptron 分类 ω_2 和 ω_3 .

2.2 问题 2

Ho-Kashyap 的算法以及 Python 代码如下所示。

Algorithm 2: Ho-Kashyap

```
1 begin initialize:  $a, b, \eta_0 < 1$ , threshold  $b_{\min, k_{\max}}$ ,  $k = 0$ ;  
2 while True do  
3    $k \leftarrow k + 1$ ;  
4    $a = a + \eta_k \sum_{y \in Y^{(k)}} y$ ;  
5    $e \leftarrow Ya - b$ ;  
6    $e^+ \leftarrow 1/2(e + \text{abs}(e))$ ;  
7    $b \leftarrow b + 2\eta_k e^+$ ;  
8    $a = Y^+b$ ;  
9   if  $(e \leq b_{\min})$  then  
10    return  $a, b$ ;  
11 print('No solutions!');
```

```
1 def Ho_Kashyap(lr=0.01, data=None, threshold=1e-10):  
2   a = np.ones([3, 1])  
3   b = np.ones([len(data), 1])  
4   data[:, 2] = 1  
5   data[10:, :] = -data[10:, :]  
6   for i in range(300000):  
7     error = data.dot(a) - b  
8     error_plus = 0.5 * (error + np.abs(error))  
9     b = b + 2 * lr * error_plus  
10    a = np.linalg.pinv(data).dot(b)  
11    if i % 100 == 0:  
12      print('iter:%d, the error is %f' % (i, error.sum()))  
13      if abs(error).sum() < threshold:  
14        print(error)  
15        return a  
16  print(error)  
17  return a
```

对 ω_1 和 ω_3 的分类结果如 Figure 5 所示。最终的 error 为: $[-7.55684779e-01, [-1.50440675e-02], [-7.65558887e-01], [-5.17913983e-01], [5.32907052e-15], [-7.55691123e-01], [-4.61518484e-01], [-4.98780350e-01], [-4.63259450e-01], [-1.98151872e-01], [-1.19612264e+00], [5.32907052e-15], [-2.06583287e+00], [-1.54708533e-01], [5.55111512e-15], [-8.43128071e-02], [1.06581410e-14], [5.55111512e-15], [5.55111512e-15], [-9.30626139e-01]]$ 。由 Figure 5 所示，该样本是线性不可分的，由误差可以看出其中存在小于 0 的元素，因此也可以说明原样本线性不可分。

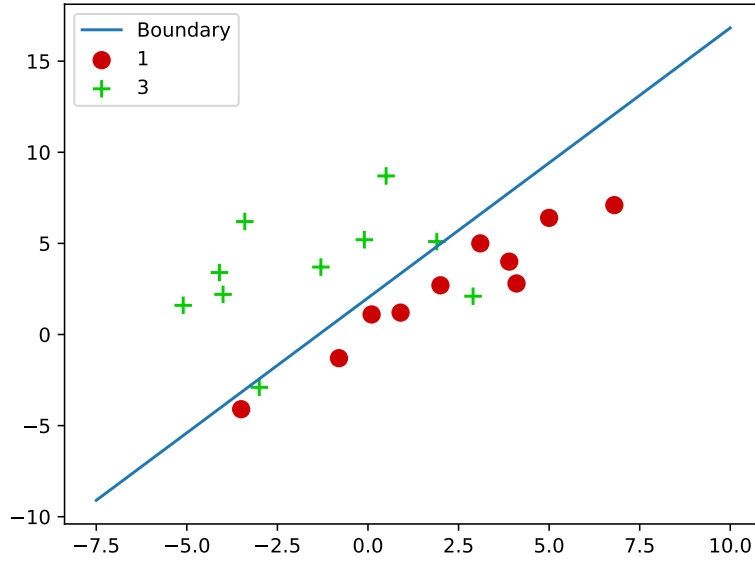


Figure 5: 利用 Ho-Kashyap 分类 ω_1 和 ω_3 .

对 ω_2 和 ω_4 的分类结果如图 Figure 6 所示。最终的 error 为 $[1.73194792e-12], [-4.66697792e-11], [1.12265752e-12], [1.41220369e-12], [1.25854882e-12], [5.65769653e-13], [5.87530025e-13], [4.10782519e-13], [1.79589676e-12], [8.99724739e-13], [1.83852933e-13], [-1.44970702e-11], [3.21520588e-13], [3.21520588e-13], [2.33146835e-13], [-2.57203148e-11], [3.42836870e-13], [6.11066753e-13], [7.58504370e-13], [5.54223334e-13]]$ 。由图 Figure 6 可以看出，该样本是线性可分的。并且 error 也全部为零（近似），因此也可以说明原样本线性可分。

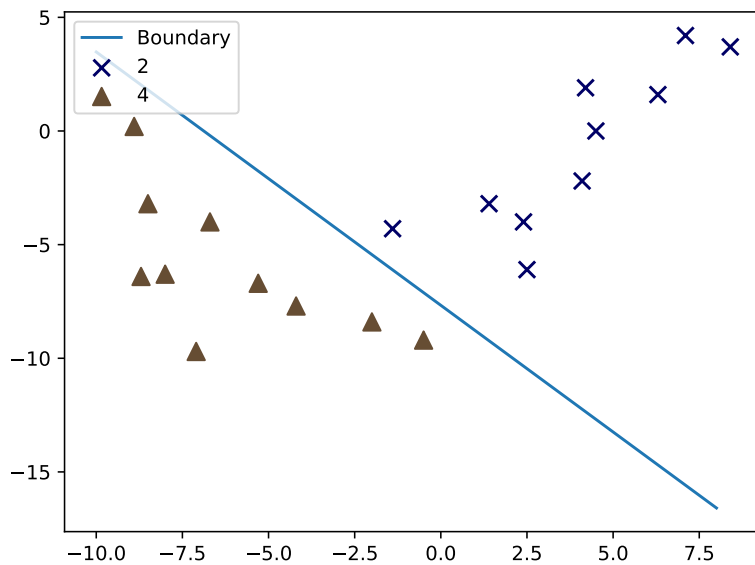


Figure 6: 利用 Ho-Kashyap 分类 ω_2 和 ω_4 .

2.3 问题 3

MSE 多类扩展算法和 Python 的代码如下所示。

Algorithm 3: MSE 多类扩展

- 1 **Objective Function:** $\min_{W,b} \sum_{i=1}^n \left\| W^T \hat{X} - Y \right\|_2^2;$
 - 2 $W = \begin{pmatrix} W \\ b^T \end{pmatrix} \in R^{(d+1) \times c}, \hat{X} = \begin{pmatrix} X \\ 1 \end{pmatrix} \in R^{d+1}, \hat{X} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n) \in R^{(d+1) \times n};$
 - 3 $\hat{W} = (\hat{X}\hat{X}^T + \lambda I)^{-1} \hat{X}Y^T \in R^{(d+1) \times c};$
-

```
1 def multi_class_mse(epsilon=0.01, data=None, label=None):  
2     W = np.linalg.inv(data.dot(data.T) + epsilon).dot(data).dot(  
        label.T)  
3     return W
```

通过利用每类前 8 个样本来构造分类器，后两个样本作测试，可以得到最终的 \hat{W} 为：

$$\hat{W} = \begin{bmatrix} 0.02049053, 0.06810353, -0.0408785, -0.04773576, \\ 0.0162581, -0.0360417, 0.05968833, -0.03991593, \\ 0.26737315, 0.27362052, 0.25018894, 0.20848962 \end{bmatrix}$$

测试的准确率为 100%。

A 附录——代码 (Python)

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4
5  def load_data(data_path):
6      data = []
7      with open(data_path) as data_file:
8          for line in data_file.readlines():
9              line_split = line.strip('\n').split(' ')
10             line_split = [float(line_split[i]) for i in range(
11                             len(line_split))]
12             data.append(line_split)
13
14     return np.array(data)
15
16 def batch_perception(lr=0.5, data=None):
17     a = np.zeros([3, 1])
18     threshold = 1e-9
19     data[:, 2] = 1
20     data[10:, :] = -data[10:, :]
21     iter = 0
22     for i in range(1000000):
23         result = a.T.dot(data.T)
24         _, inx = np.where(result <= 0)
25         increments = np.sum(lr * data[inx], axis=0)
26         a += increments.reshape([3, 1])
27         iter += 1
28         if abs(increments.sum()) < threshold:
29             print('total iter:%d' % iter)
30             return a
31
32 def Ho_Kashyap(lr=0.01, data=None, threshold=1e-10):
33     a = np.ones([3, 1])
```



```

34     b = np.ones([len(data), 1])
35     data[:, 2] = 1
36     data[10:, :] = -data[10:, :]
37     for i in range(300000):
38         error = data.dot(a) - b
39         error_plus = 0.5 * (error + np.abs(error))
40         b = b + 2 * lr * error_plus
41         a = np.linalg.pinv(data).dot(b)
42         if i % 100 == 0:
43             print('iter:%d, the error is %f' % (i, error.sum()))
44             if abs(error).sum() < threshold:
45                 print(error)
46                 return a
47     print(error)
48     return a
49
50
51 def multi_class_mse(epsilon=0.01, data=None, label=None):
52     W = np.linalg.inv(data.dot(data.T) + epsilon).dot(data).dot
53         (label.T)
54     return W
55
56 if __name__ == '__main__':
57     data = load_data('data.txt')
58     # show the data
59     plt.figure()
60     plt.scatter(data[:10, 0], data[:10, 1], marker='o', label='
        1', color=(0.8, 0., 0.), s=70)
61     plt.scatter(data[10:20, 0], data[10:20, 1], marker='x',
62         label='2', color=(0., 0., 0.4), s=70)
63     plt.scatter(data[20:30, 0], data[20:30, 1], marker='+',
64         label='3', color=(0., 0.8, 0.), s=80)
65     plt.scatter(data[30:40, 0], data[30:40, 1], marker='^',
66         label='4', color=(0.4, 0.3, 0.2), s=70)

```

```

64 plt.legend(loc='upper left')
65
66 #####batch_perception#####
67 # train data of w1 and w2
68 data_temp = np.zeros([20, 3])
69 data_temp[:10] = data[:10]
70 data_temp[10:20] = data[10:20]
71 a = batch_perception(data=data_temp)
72 a = a / a[1]
73 a_x = np.linspace(-7.5, 10, 10)
74 a_y = -a_x * a[0] - a[2]
75 plt.figure()
76 plt.scatter(data[:10, 0], data[:10, 1], marker='o', label='
    1', color=(0.8, 0., 0.), s=70)
77 plt.scatter(data[10:20, 0], data[10:20, 1], marker='x',
    label='2', color=(0., 0., 0.4), s=70)
78 plt.plot(a_x, a_y, '-', label='Boundary')
79 plt.legend(loc='upper left')
80
81 # train data of w2 and w3
82 data_temp = np.zeros([20, 3])
83 data_temp[:10] = data[10:20]
84 data_temp[10:20] = data[20:30]
85 a = batch_perception(data=data_temp)
86 a = a / a[1]
87 a_x = np.linspace(-7.5, 10, 10)
88 a_y = -a_x * a[0] - a[2]
89 plt.figure()
90 plt.scatter(data[10:20, 0], data[10:20, 1], marker='x',
    label='2', color=(0., 0., 0.4), s=70)
91 plt.scatter(data[20:30, 0], data[20:30, 1], marker='+',
    label='3', color=(0., 0.8, 0.), s=80)
92 plt.plot(a_x, a_y, '-', label='Boundary')
93 plt.legend(loc='upper right')
94
95 #####Ho_Kashyap#####

```

```

96     # train data of w1 and w3
97     data_temp = np.zeros([20, 3])
98     data_temp[:10] = data[:10]
99     data_temp[10:20] = data[20:30]
100    a = Ho_Kashyap(data=data_temp, threshold=1)
101    a = a / a[1]
102    a_x = np.linspace(-7.5, 10, 10)
103    a_y = - a_x * a[0] - a[2]
104    plt.figure()
105    plt.scatter(data[:10, 0], data[:10, 1], marker='o', label='
        1', color=(0.8, 0., 0.), s=70)
106    plt.scatter(data[20:30, 0], data[20:30, 1], marker='+',
        label='3', color=(0., 0.8, 0.), s=80)
107    plt.plot(a_x, a_y, '-', label='Boundary')
108    plt.legend(loc='upper left')
109
110    # train data of w2 and w4
111    data_temp = np.zeros([20, 3])
112    data_temp[:10] = data[10:20]
113    data_temp[10:20] = data[30:40]
114    a = Ho_Kashyap(data=data_temp)
115    a = a / a[1]
116    a_x = np.linspace(-10, 8, 10)
117    a_y = - a_x * a[0] - a[2]
118    plt.figure()
119    plt.scatter(data[10:20, 0], data[10:20, 1], marker='x',
        label='2', color=(0., 0., 0.4), s=70)
120    plt.scatter(data[30:40, 0], data[30:40, 1], marker='^',
        label='4', color=(0.4, 0.3, 0.2), s=80)
121    plt.plot(a_x, a_y, '-', label='Boundary')
122    plt.legend(loc='upper left')
123
124    #####Multi-Classification#####
125    train_data = np.zeros([3, 32])
126    train_data[:, :8] = data[:8].T
127    train_data[:, 8:16] = data[10:18].T

```

```

128     train_data[:, 16:24] = data[20:28].T
129     train_data[:, 24:32] = data[30:38].T
130     train_data[2, :] = 1
131
132     train_label = np.zeros([4, 32])
133     train_label[0, :8] = 1
134     train_label[1, 8:16] = 1
135     train_label[2, 16:24] = 1
136     train_label[3, 24:32] = 1
137
138     W = multi_class_mse(data=train_data, label=train_label)
139
140     test_data = np.zeros([3, 8])
141     test_data[:, :2] = data[8:10].T
142     test_data[:, 2:4] = data[18:20].T
143     test_data[:, 4:6] = data[28:30].T
144     test_data[:, 6:8] = data[38:40].T
145     test_data[2, :] = 1
146
147     test_label = np.zeros([4, 8])
148     test_label[0, :2] = 1
149     test_label[1, 2:4] = 1
150     test_label[2, 4:6] = 1
151     test_label[3, 6:8] = 1
152
153     predict = W.T.dot(test_data)
154     inx = np.argmax(predict, axis=0)
155     cnt = 0
156     for i in range(8):
157         if test_label[inx[i], i] == 1:
158             cnt += 1
159     acc = cnt / test_label.shape[1]
160     print('The accuracy is: %f' % acc)
161
162     plt.show()

```