第三次作业

姓名：樊宇

学号：2021E8018782022

Part 1

Question 1

规范化增广样本：

$$y_1 = \begin{bmatrix}1\\4\\1\end{bmatrix} \quad y_2 = \begin{bmatrix}2\\3\\1\end{bmatrix} \quad y_3 = \begin{bmatrix}-4\\-1\\-1\end{bmatrix} \quad y_4 = \begin{bmatrix}-3\\-2\\-1\end{bmatrix}$$

iteration 1：

错分样本：

$$y_3 \quad y_4$$

批处理感知准则函数：

$$\begin{aligned} a &= a + \eta_k \sum_{y \in Y(k)} y \\ &= \begin{bmatrix}0\\1\\0\end{bmatrix} + \begin{bmatrix}-4\\-1\\-1\end{bmatrix} + \begin{bmatrix}-3\\-2\\-1\end{bmatrix} \\ &= \begin{bmatrix}-7\\-2\\-2\end{bmatrix} \end{aligned}$$

iteration 2：

错分样本：

$$y_1 \quad y_2$$

批处理感知准则函数：

$$\begin{aligned} a &= a + \eta_k \sum_{y \in Y(k)} y \\ &= \begin{bmatrix}-7\\-2\\-2\end{bmatrix} + \begin{bmatrix}1\\4\\1\end{bmatrix} + \begin{bmatrix}2\\3\\1\end{bmatrix} \\ &= \begin{bmatrix}-4\\5\\0\end{bmatrix} \end{aligned}$$

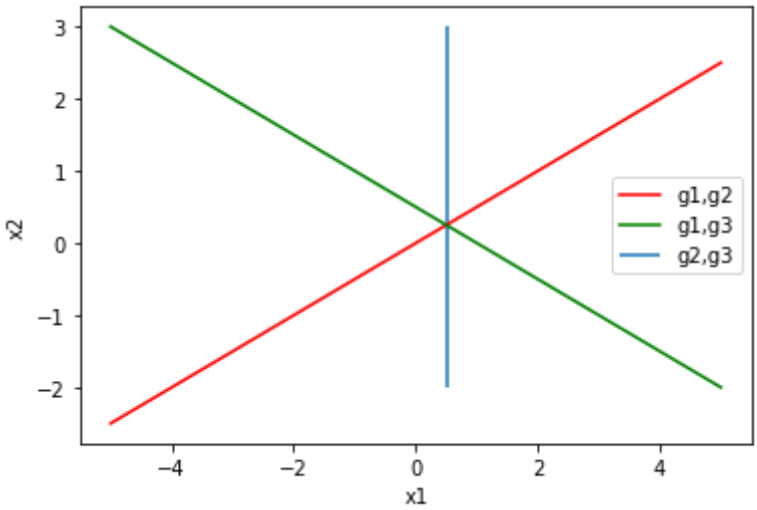
验证得：

$$\forall y_k \quad g(y_k) > 0$$

即：

$$a = \begin{bmatrix} -4 \\ 5 \\ 0 \end{bmatrix}$$

Question 2



决策面分别为：

$$g1(\boldsymbol{x}) - g2(\boldsymbol{x})$$
$$g1(\boldsymbol{x}) - g3(\boldsymbol{x})$$
$$g2(\boldsymbol{x}) - g3(\boldsymbol{x})$$

由于决策面恰好交于一点，因此决策面不存在不确定区域

Part 2

Question 1

(a)

$$a = [34, 30.39999815, 34.10000217]^T$$
$$iteration = 24$$

(b)

$$a = [31, 5.09999611, 5.19999525]^T$$
$$iteration = 40$$

Question 2

w_1 **和** w_3

a =

```
array([[ 1.29562105],
       [ 1.45232755],
       [-0.37736604]])
```

b =

```
array([[ 1.62840941],
       [10.02521215],
       [ 0.65860439],
       [ 3.24874475],
       [ 6.19353899],
       [ 5.66453021],
       [ 0.68566603],
       [ 2.57118602],
       [ 8.24304928],
       [ 5.57056993],
       [ 3.18538359],
       [ 1.26129962],
       [ 0.63775602],
       [ 0.97789079],
       [ 5.34389446],
       [ 1.98865905],
       [ 5.98196212],
       [ 5.94196634],
       [ 6.71503499],
       [ 0.69194551]])
```

结果为：

```
array([[ 1.02575116],
       [ 8.49214985],
       [-2.24032467],
       [ 3.18138783],
       [ 6.19353899],
       [ 3.91100614],
       [ 0.62433482],
       [ 2.14987655],
       [ 6.14211613],
       [ 5.45023449],
       [ 1.96700006],
       [ 1.26129962],
       [-4.71490245],
       [ 0.81191503],
       [ 5.34389446],
       [ 1.98865905],
       [ 5.98196212],
       [ 5.94196634],
       [ 6.71503499],
       [-2.13047661]])
```

迭代步长为10000次

正确率为：85%

分析：

可能这两类样本不是线性可分的

w_2**和**w_4

a =

```
array([[8.92591648],
       [1.60797963],
       [1.49750589]])
```

b =

```
array([[26.63178996],
       [ 0.24377442],
       [16.16162485],
       [21.45194448],
       [18.52446747],
       [ 6.38496795],
       [ 6.79493935],
       [ 3.81100556],
       [27.97339722],
       [12.22396001],
       [ 6.86905831],
       [ 5.08818093],
       [ 9.35833475],
       [ 9.53387131],
       [ 7.83752895],
       [ 5.65970271],
       [ 9.62960475],
       [14.64743352],
       [17.01641033],
       [13.37211042]])
```

结果为：

```
array([[26.63209613],
       [ 0.23546942],
       [16.1618248 ],
       [21.45219789],
       [18.52469176],
       [ 6.38506899],
       [ 6.79504416],
       [ 3.81107974],
       [27.9737166 ],
       [12.22411976],
       [ 6.8690917 ],
       [ 5.08560038],
       [ 9.35839273],
       [ 9.53392926],
       [ 7.83757027],
       [ 5.65512726],
       [ 9.62966504],
       [14.64754381],
       [17.01654558],
       [13.37220793]])
```

迭代步长为：629

正确类为：100%

分析：

这两类样本之间是线性可分的

Question 3

W =

```
array([[ 0.26747287,  0.27372075,  0.25027714,  0.20852924],
       [ 0.02049668,  0.06810971, -0.04087307, -0.04773332],
       [ 0.01626151, -0.03603827,  0.05969134, -0.03991458]])
```

正确率为： 100%

```
import numpy as np
```

```
data = np.zeros(shape=(4, 10, 3))
```

```
data1 = np.array([[0.1, 1.1], [6.8, 7.1], [-3.5, -4.1], [2., 2.7], [4.1, 2.8],
[3.1, 5.], [-0.8, -1.3], [0.9, 1.2], [5., 6.4], [3.9, 4.]], dtype = np.float32)
data1 = np.c_[np.ones(shape=(10, 1)), data1]
data2 = np.array([[7.1, 4.2], [-1.4, -4.3], [4.5, 0.], [6.3, 1.6], [4.2, 1.9],
[1.4, -3.2], [2.4, -4.], [2.5, -6.1], [8.4, 3.7], [4.1, -2.2]], dtype =
np.float32)
data2 = np.c_[np.ones(shape=(10, 1)), data2]
data3 = np.array([[-3., -2.9], [0.5, 8.7], [2.9, 2.1], [-0.1, 5.2], [-4., 2.2],
[-1.3, 3.7], [-3.4, 6.2], [-4.1, 3.4], [-5.1, 1.6], [1.9, 5.1]], dtype =
np.float32)
data3 = np.c_[np.ones(shape=(10, 1)), data3]
data4 = np.array([[-2., -8.4], [-8.9, 0.2], [-4.2, -7.7], [-8.5, -3.2], [-6.7,
-4.], [-0.5, -9.2], [-5.3, -6.7], [-8.7, -6.4], [-7.1, -9.7], [-8., -6.3]],
dtype = np.float32)
data4 = np.c_[np.ones(shape=(10, 1)), data4]
```

```
for i in range(4):
    data[i] = eval('data'+str(i+1))
np.save('data.npy', data)
```

```
def batch_perception(data, eta, rho, threshold):
    a = np.zeros(shape = (3, 1))
    iteration = 0
    while True:
        iteration += 1
        Y_k = []
        for i in range(data.shape[0]):
            y = data[i, :]
            y = y.reshape(1, 3)
            if y.dot(a) <= 0:
                Y_k.append(y.tolist())
        t = eta * np.array(Y_k).sum(axis = 0)
        a = a + eta * t.T
        if len(Y_k) == 0 or np.linalg.norm(t, ord = 1, axis =1) < threshold:
            return a, iteration
        else:
            eta = eta * rho
```

w_1和w_2

```
# 规范化增广样本
data_12 = np.r_[data[0], -data[1]]
```

```
a, iteration = batch_perception(data_12, 1, 1, 0.01)
```

```
a
```

```
array([[ 34.        ],
       [-30.39999815],
       [ 34.10000217]])
```

```
iteration
```

```
24
```

```
data_12.dot(a)
```

```
array([[ 68.47000334],
       [ 69.39001893],
       [  0.58998788],
       [ 65.27001118],
       [  4.84001493],
       [110.26001948],
       [ 13.98999769],
       [ 47.56000662],
       [100.24002638],
       [ 51.84001299],
       [ 38.61998136],
       [ 70.07001915],
       [102.79999168],
       [102.95998986],
       [ 28.88998313],
       [117.68000526],
       [175.36000714],
       [250.01000536],
       [ 95.18996322],
       [165.65999592]])
```

w_3和w_4

```
# 规范化增广样本
data_34 = np.r_[data[2], -data[3]]
```

```
a, iteration = batch_perception(data_34, 1, 1, 0.01)
```

```
a
```

```
array([[31.        ],
       [ 5.09999611],
       [ 5.19999525]])
```

```
iteration
```

```
40
```

```
data_34.dot(a)
```

```
array([[ 0.62002494],
       [78.78995574],
       [56.70997874],
       [57.52997469],
       [22.04000535],
       [43.60998797],
       [45.89998229],
       [27.77000077],
       [13.31001284],
       [67.20996777],
       [22.87995034],
       [13.34996439],
       [30.45994513],
       [28.989952  ],
       [23.96995398],
       [19.38995336],
       [30.86994755],
       [46.6499353 ],
       [55.64992484],
       [42.55993996]])
```

```
import numpy as np
data = np.load('data.npy')
```

```
def Ho_Kashyap(data, k_max, threshold, eta, rho):
    iteration = 0
    a = np.random.rand(3, 1)
    b = np.random.rand(data.shape[0], 1)
    while iteration <= k_max:
        iteration += 1
        e = data.dot(a) - b
        e_ = 0.5 * (e + abs(e))
        b = b + 2 * eta * e_
        a = np.linalg.pinv(data.T.dot(data)).dot(data.T).dot(b)
        if np.linalg.norm(e) < threshold:
            return a, b, iteration
        if iteration == k_max:
            print('no solution found')
            return a, b, iteration
        eta = eta * rho
```

```
data_1 = np.r_[data[0], -data[2]]
a_1, b_1, iteration_1 = Ho_Kashyap(data_1, 10000, 0.01, 0.9, 1)
```

```
no solution found
```

```
a_1
```

```
array([[ 1.29562105],
       [ 1.45232755],
       [-0.37736604]])
```

```
b_1
```

```
array([[ 1.62840941],
       [10.02521215],
       [ 0.65860439],
       [ 3.24874475],
       [ 6.19353899],
       [ 5.66453021],
       [ 0.68566603],
       [ 2.57118602],
       [ 8.24304928],
```

```
       [ 5.57056993],
       [ 3.18538359],
       [ 1.26129962],
       [ 0.63775602],
       [ 0.97789079],
       [ 5.34389446],
       [ 1.98865905],
       [ 5.98196212],
       [ 5.94196634],
       [ 6.71503499],
       [ 0.69194551]])
```

```
iteration_1
```

```
10000
```

```
result_1 = data_1.dot(a_1)
result_1
```

```
array([[ 1.02575116],
       [ 8.49214985],
       [-2.24032467],
       [ 3.18138783],
       [ 6.19353899],
       [ 3.91100614],
       [ 0.62433482],
       [ 2.14987655],
       [ 6.14211613],
       [ 5.45023449],
       [ 1.96700006],
       [ 1.26129962],
       [-4.71490245],
       [ 0.81191503],
       [ 5.34389446],
       [ 1.98865905],
       [ 5.98196212],
       [ 5.94196634],
       [ 6.71503499],
       [-2.13047661]])
```

```
err_rate_1 = np.sum(result_1 < 0) / result_1.shape[0]
err_rate_1
```

```
0.15
```

```
data_2 = np.r_[data[1], -data[3]]
a_2, b_2, iteration_2 = Ho_Kashyap(data_2, 10000, 0.01, 0.9, 1)
```

```
a_2
```

```
array([[8.92591648],
       [1.60797963],
       [1.49750589]])
```

```
b_2
```

```
array([[26.63178996],
       [ 0.24377442],
       [16.16162485],
       [21.45194448],
       [18.52446747],
       [ 6.38496795],
       [ 6.79493935],
       [ 3.81100556],
       [27.97339722],
       [12.22396001],
       [ 6.86905831],
       [ 5.08818093],
       [ 9.35833475],
       [ 9.53387131],
       [ 7.83752895],
       [ 5.65970271],
       [ 9.62960475],
       [14.64743352],
       [17.01641033],
       [13.37211042]])
```

```
iteration_2
```

```
629
```

```
result_2 = data_2.dot(a_2)
result_2
```

```
array([[26.63209613],
       [ 0.23546942],
       [16.1618248 ],
       [21.45219789],
       [18.52469176],
       [ 6.38506899],
       [ 6.79504416],
       [ 3.81107974],
       [27.9737166 ],
       [12.22411976],
       [ 6.8690917 ],
       [ 5.08560038],
       [ 9.35839273],
       [ 9.53392926],
       [ 7.83757027],
       [ 5.65512726],
       [ 9.62966504],
       [14.64754381],
       [17.01654558],
       [13.37220793]])
```

```
err_rate_2 = np.sum(result_2 < 0) / result_2.shape[0]
err_rate_2
```

```
0.0
```

```

```

```
import numpy as np
import math
data = np.load('data.npy')
```

```
data_train = np.zeros(shape = (32, 3))
data_test = np.zeros(shape = (8, 3))
```

```
for i in range(4):
    data_train[ 8 * i: 8 * (i + 1), :] = data[i, 0 : 8, :]
    data_test[2 * i: 2 * (i + 1), :] = data[i, -2:, :]
```

```
label_1 = np.array([1, 0, 0, 0])
label_2 = np.array([0, 1, 0, 0])
label_3 = np.array([0, 0, 1, 0])
label_4 = np.array([0, 0, 0, 1])
```

```
labels_train = np.zeros(shape = (32, 4))
labels_test = np.zeros(shape = (8, 4))
```

```
for i in range(4):
    labels_train[ 8 * i: 8 * (i + 1), :] = eval('label_' + str(i + 1))
    labels_test[ 2 * i: 2 * (i + 1), :] = eval('label_' + str(i + 1))
```

```
W =
np.linalg.pinv(data_train.T.dot(data_train)).dot(data_train.T).dot(labels_train)
W
```

```
array([[ 0.26747287,  0.27372075,  0.25027714,  0.20852924],
       [ 0.02049668,  0.06810971, -0.04087307, -0.04773332],
       [ 0.01626151, -0.03603827,  0.05969134, -0.03991458]])
```

```
test = data_test.dot(W)
test
```

```
array([[ 0.47402991,  0.38362434,  0.42793641, -0.28559066],
       [ 0.41245595,  0.39519553,  0.32963755, -0.13728902],
       [ 0.49981253,  0.71250067,  0.12780137, -0.34011457],
       [ 0.31573393,  0.63225475, -0.04862338,  0.1006347 ],
       [ 0.18895823, -0.13130001,  0.55423592,  0.38810585],
       [ 0.38935025,  0.219334  ,  0.47704416, -0.08572841],
       [-0.03579016,  0.13971306, -0.03853011,  0.93460721],
       [ 0.00105195, -0.0441158 ,  0.2012062 ,  0.84185766]])
```

```
index = np.argmax(test, axis = 1)
```

```
index
```

```
array([0, 0, 1, 1, 2, 2, 3, 3], dtype=int64)
```

```
for i in range(test.shape[0]):
    test[i, index[i]] = 1
```

```
test[test < 1] = 0
```

```
test
```

```
array([[1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 0., 1.]])
```

```
err = 0
```

```
for i in range(8):
    if test[i, :].all() != labels_test[i, :].all():
        err += 1
```

```
correct_rate = (8 - err) / 8
correct_rate
```

```
1.0
```