Homework 1

Jingwei Zhang 201528013229095

2015-11-16

1 Problem 1

1. 进行规范化增广处理.

有样本: $y_1=\begin{bmatrix}1\\1\\4\end{bmatrix}$, $y_2=\begin{bmatrix}1\\2\\3\end{bmatrix}$, $y_3=\begin{bmatrix}-1\\-4\\-1\end{bmatrix}$, $y_4=\begin{bmatrix}-1\\-3\\-2\end{bmatrix}$; $a_0=\begin{bmatrix}0\\0\\1\end{bmatrix}$ $\eta_k\equiv 1$

对 $k=1$, $a_0^T y_i>0$, $i=1,2$; $Y_0=\{y_3,y_4\}$.

有 $a_1=a_0+\eta_k\sum_{y\in Y_0} y=\begin{bmatrix}0\\0\\1\end{bmatrix}+\begin{bmatrix}-2\\-7\\-3\end{bmatrix}=\begin{bmatrix}-2\\-7\\-2\end{bmatrix}$

$k=2$, $a_1^T y_i>0$, $i=3,4$, $Y_1=\{y_1,y_2\}$.

有 $a_2=a_1+\eta_k\sum_{y\in Y_1} y=\begin{bmatrix}-2\\-7\\-2\end{bmatrix}+\begin{bmatrix}2\\3\\7\end{bmatrix}=\begin{bmatrix}0\\-5\\5\end{bmatrix}$

又 $\forall i=1,\dots,4$, $a_2^T y_i>0$, 所有样本均正确分类, 停止

$\therefore a=a_2=\begin{bmatrix}0\\-5\\5\end{bmatrix}$

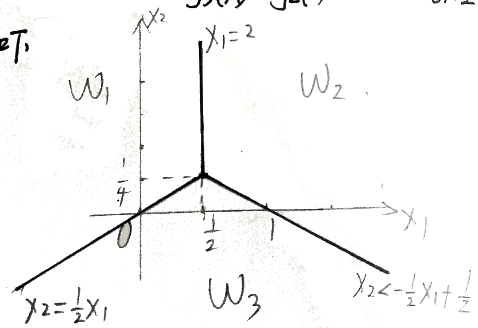
2 Problem 2

2 ①当被分为第1类时，有 $\begin{cases} g_1(x)>g_2(x) \\ g_1(x)>g_3(x) \end{cases}$ 即 $\begin{cases} -x_1+x_2>-x_2 \\ -x_1+x_2>x_1+x_2-1 \end{cases}$，得 ω_1 的区域为 $\begin{cases} x_2>\frac{1}{2}x_1 \\ x_1<\frac{1}{2} \end{cases}$

②当被分为第2类时，有 $\begin{cases} g_2(x)>g_1(x) \\ g_2(x)>g_3(x) \end{cases}$ 易得 $\begin{cases} x_2>-\frac{1}{2}x_1+\frac{1}{2} \\ x_1>\frac{1}{2} \end{cases}$

③当被分为第3类时，有 $\begin{cases} g_3(x)>g_1(x) \\ g_3(x)>g_2(x) \end{cases}$ 易得 $\begin{cases} x_2<\frac{1}{2}x_1 \\ x_2<-\frac{1}{2}x_1+\frac{1}{2} \end{cases}$，

绘图如下



由于在 R^2 上任意一点，$g_i(x)$，$i=1,2,3$ 都会有一个最大值，同时判决函数互不相同，故最大值有多个的情况只会发生在判决面上，故不存在分类不定的区域

3 Problem 3

3. 由题意，易得增广的样本阵为

$$Y=\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{bmatrix} \quad \text{其伪逆为} \quad Y^+=\begin{bmatrix} \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

又 $\eta_k\equiv1$，$b_0=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$

① 由 $b_0=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$ 得 $a_0=Y^+b_0=\begin{bmatrix}0\\0\\0\end{bmatrix}$

②

$e_1=Ya-b=\begin{bmatrix}-1\\-1\\-1\\-1\end{bmatrix}$ 故 $e_1^+=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$

$\therefore b_1=b_0+2\eta_k\cdot e^+=b_0=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$

$\therefore b$ 收敛于 $b_0=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$，$a$ 收敛于 $a_0=\begin{bmatrix}0\\0\\0\end{bmatrix}$，$e$ 收敛于 $\begin{bmatrix}-1\\-1\\-1\\-1\end{bmatrix}$

因 e_1 各分量均为负，样本不是线性可分的

4 Problem 4

4.a) 令 x_p 为 x_a 在 $g(x)=0$ 上的投影，

则 $x_a = x_p + r\frac{w}{||w||}$ （$r>0$ 时，x_a 在 $g(x)=0$ 正向，$r<0$ 时 x_a 在 $g(x)=0$ 负向）

对 $g(x)=0$ 上任意一点 x，令 $b = x - x_p$，又 $g(x)=0$，$g(x_p)=0$，有 $w^T \cdot b = 0$.

将 x 表示成 $x_a - r\frac{w}{||w||} + b$.

则最优化问题转化为：$\min ||-r\frac{w}{||w||}+b||^2$

$$\text{s.t. } g(x_a - r\frac{w}{||w||} + b) = 0.$$

又 $||-r\frac{w}{||w||}+b||^2 = r^2 + ||b||^2 - 2r\frac{w^T}{||w||}\cdot b = r^2 + ||b||^2 \geq r^2$，当 $b=0$ 时等号成立.

$\therefore x_p$ 是 $g(x)=0$ 上距离 x_a 最近的点，即 $||x_a - x_p|| = ||r\frac{w}{||w||}|| = r$ 为 x_a 到 $g(x)=0$ 的距离

又 $x_p = x_a - r\frac{w}{||w||}$ 在 $g(x)=0$ 上，有：

$$w^T(x_a - r\frac{w}{||w||}) + w_0 = w^T x_a + w_0 - r||w|| = g(x_a) - r||w|| = 0$$

得 $r = \frac{g(x_a)}{||w||}$

$\therefore x_a$ 到超平面 $g(x)=0$ 距离为 $|r| = \frac{|g(x_a)|}{||w||}$

b)：将 $r = \frac{g(x_a)}{||w||}$ 代入 $x_a = x_p + r\frac{w}{||w||}$

得 $x_p = x_a - \frac{g(x_a)}{||w||^2}\cdot w$

5 Programming Problem 1

5.1 Result

The Number of Iterations: 23 for ω_1 and ω_2, 16 for ω_3 and ω_2. From figures blow we can find that points of ω_3 and ω_2 are much more scattered, so that $\sum_{y\in Y} y$ will be farther away from current a. This will make a converge quicker.

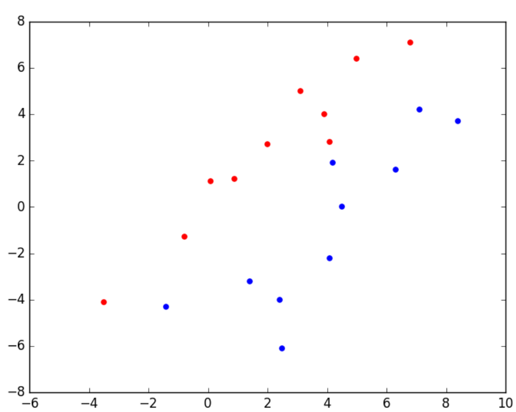


Figure 1: Figure for samples belonging to ω_1 and ω_2

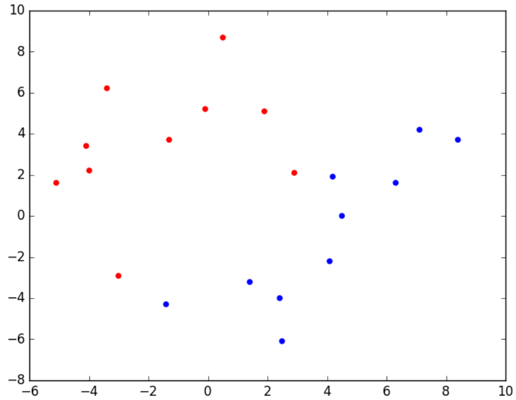


Figure 2: Figure for samples belonging to ω_3 and ω_2

5.2 Code

```
#!/usr/bin/python3
# coding=utf-8

import numpy as np
import matplotlib.pyplot as plt

# constant values
omega1 = np.array([
                    [0.1, 1.1],
                    [6.8, 7.1],
                    [-3.5, -4.1],
                    [2.0, 2.7],
                    [4.1, 2.8],
                    [3.1, 5.0],
                    [-0.8, -1.3],
                    [0.9, 1.2],
                    [5.0, 6.4],
                    [3.9, 4.0]
                    ])
omega2 = np.array([
                    [7.1, 4.2],
                    [-1.4, -4.3],
                    [4.5, 0.0],
                    [6.3, 1.6],
                    [4.2, 1.9],
                    [1.4, -3.2],
                    [2.4, -4.0],
                    [2.5, -6.1],
                    [8.4, 3.7],
                    [4.1, -2.2]
                    ])
omega3 = np.array([
                    [-3.0, -2.9],
                    [0.5, 8.7],
                    [2.9, 2.1],
                    [-0.1, 5.2],
                    [-4.0, 2.2],
```

```
                    [-1.3, 3.7],
                    [-3.4, 6.2],
                    [-4.1, 3.4],
                    [-5.1, 1.6],
                    [1.9, 5.1]
                    ])
origin = [0,0,0]

def eta(k):
    return 1

def batch_perception(a, b, start_a):
    s = [[1,x[0],x[1]] for x in a]
    s.extend([[-1,-x[0],-x[1]] for x in b])
    s = np.array(s)
    a = start_a
    Y = s[np.inner(a,s) <= 0]
    iter_cnt = 0
    while(len(Y) > 0):
        iter_cnt += 1
        a += eta(iter_cnt) * np.sum(Y, axis=0)
        Y = s[np.inner(a,s) <= 0]
    return a,iter_cnt

if __name__ == '__main__':
    # Problem a)
    a_a, cnt_a = batch_perception(omega1, omega2, origin)
    print(a_a,cnt_a)

    # Problem b)
    a_b, cnt_b = batch_perception(omega3, omega2, origin)
    print(a_b,cnt_b)
```

6 Programming Problem 2

6.1 Result

Analyses: The two problems are both non-linear separable. From the program we get that all elements of e is less than 0 when it ends. The convergence rate of ω_2 and ω_4 is faster than that of ω_1 and ω_3.

6.2 Code

```
#!/usr/bin/python3
# coding=utf-8

import numpy as np
import matplotlib.pyplot as plt

# constant values
omega1 = np.array([
                    [0.1, 1.1],
                    [6.8, 7.1],
                    [-3.5, -4.1],
                    [2.0, 2.7],
                    [4.1, 2.8],
                    [3.1, 5.0],
                    [-0.8, -1.3],
                    [0.9, 1.2],
                    [5.0, 6.4],
                    [3.9, 4.0]
                    ])
omega2 = np.array([
                    [7.1, 4.2],
                    [-1.4, -4.3],
                    [4.5, 0.0],
                    [6.3, 1.6],
                    [4.2, 1.9],
                    [1.4, -3.2],
                    [2.4, -4.0],
                    [2.5, -6.1],
                    [8.4, 3.7],
                    [4.1, -2.2]
                    ])
omega3 = np.array([
                    [-3.0, -2.9],
                    [0.5, 8.7],
                    [2.9, 2.1],
                    [-0.1, 5.2],
                    [-4.0, 2.2],
                    [-1.3, 3.7],
                    [-3.4, 6.2],
                    [-4.1, 3.4],
                    [-5.1, 1.6],
                    [1.9, 5.1]
                    ])
omega4 = np.array([
                    [-2.0, 8.4],
                    [-8.9, 0.2],
                    [-4.2, 7.7],
                    [-8.5, -3.2],
                    [-6.7, -4.0],
                    [-0.5, -9.2],
                    [-5.3, -6.7],
```

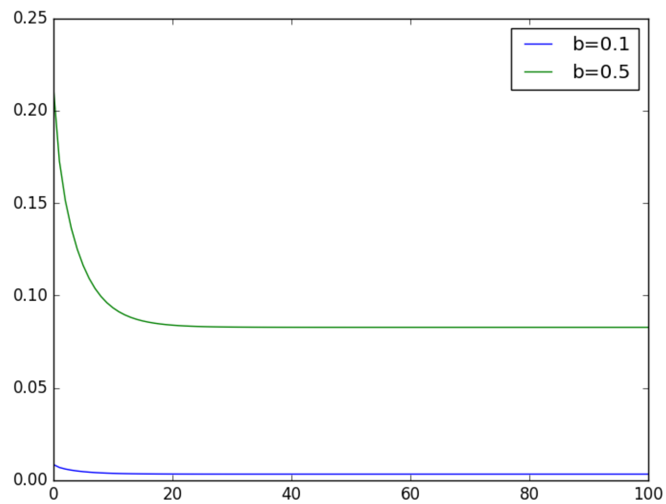
```
                    [-8.7,  -6.4],
                    [-7.1,  -9.7],
                    [-8.0,  -6.3]
                    ])
origin = [0,0,0]

def eta(k):
    return 0.5

def rand_start(n):
    return [np.random.uniform() for i in range(n)]

def get_init_b(n):
    return [np.random.uniform() for i in range(n)]

def Ho_Kashyap(s1, s2, a_start, b_start, k_max, b_min):
    s = [[1,x[0],x[1]] for x in s1]
    s.extend([[-1,-x[0],-x[1]] for x in s2])
    Y = np.matrix(s)
    Y_inv = np.linalg.pinv(Y)
    a = np.matrix(a_start).T
    b = np.matrix(b_start).T
    all_e = []
    split_able = False
    for k in range(1, k_max+1):
        # inpu = input("interation")
        e = Y * a - b
        # all_e.append((e.T * e)[0,0])
        cnt = 0
        for i in range(len(s)):
            if (a.T * ((Y[i,]).T))[0, 0] < 0:
                cnt += 1
        all_e.append(cnt)
        # print(cnt)
        if(np.max(np.abs(e)) < b_min):
            split_able = True
            break
        e_plus = e
        e_plus[e < 0] = 0
        b = b + (2 * eta(k)) * e_plus
        a = Y_inv * b
    print(split_able)
    return all_e

if __name__ == '__main__':
    k_max = 10000
    start_a = rand_start(3)
    start_b = get_init_b(len(omega1)+len(omega3))
    b_min = 1E-5
    # omega 1 and omega 3
    x = [i for i in range(1, k_max + 1)]
    all_E_1 = Ho_Kashyap(omega1, omega3,
        start_a, start_b, k_max, b_min)
    # print(all_E_1[len(all_E_1)-1])
    # omega 2 and omega 4
    all_E_2 = Ho_Kashyap(omega2, omega4,
```

```
    start_a , start_b , k_max, b_min)
# print(all_E_2)
# plot
ax = plt.gca()
ax.set_xscale('log')
plt.title("Training Errors")
plt.plot(x, all_E_1 , color='b',
    label="omega 1 and omega 3")
plt.plot(x, all_E_2 , color='r',
    label="omega 2 and omega 4")
plt.legend()
plt.show()
```

7 Programming Problem 3

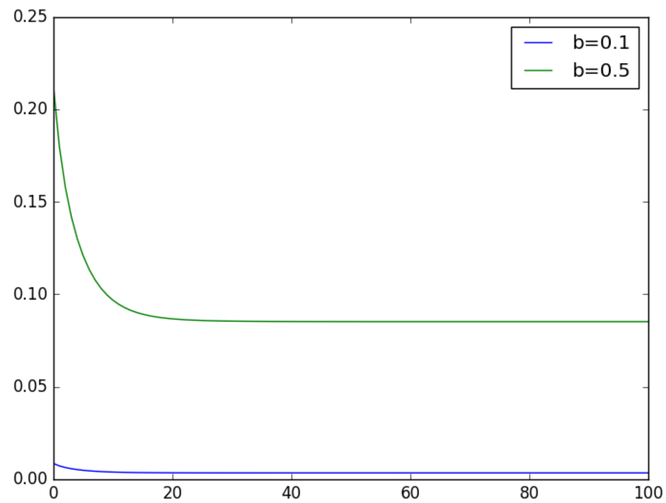
7.1 Result

Batch Relaxation:



Convergence Rate: The convergence rate of $b = 0.1$ is a bit faster than that of $b = 0.5$.

Single Sample Relaxation:



7.2 Code

```
#!/usr/bin/python3
# coding=utf-8

import numpy as np
import matplotlib.pyplot as plt

# constant values
omega1 = np.array([
    [0.1, 1.1],
    [6.8, 7.1],
    [-3.5, -4.1],
    [2.0, 2.7],
    [4.1, 2.8],
    [3.1, 5.0],
    [-0.8, -1.3],
    [0.9, 1.2],
    [5.0, 6.4],
    [3.9, 4.0]
])
omega2 = np.array([
    [7.1, 4.2],
    [-1.4, -4.3],
    [4.5, 0.0],
    [6.3, 1.6],
    [4.2, 1.9],
    [1.4, -3.2],
    [2.4, -4.0],
    [2.5, -6.1],
    [8.4, 3.7],
    [4.1, -2.2]
])
omega3 = np.array([
    [-3.0, -2.9],
    [0.5, 8.7],
    [2.9, 2.1],
```

```
    [-0.1, 5.2],
    [-4.0, 2.2],
    [-1.3, 3.7],
    [-3.4, 6.2],
    [-4.1, 3.4],
    [-5.1, 1.6],
    [1.9, 5.1]
])
omega4 = np.array([
    [-2.0, 8.4],
    [-8.9, 0.2],
    [-4.2, 7.7],
    [-8.5, -3.2],
    [-6.7, -4.0],
    [-0.5, -9.2],
    [-5.3, -6.7],
    [-8.7, -6.4],
    [-7.1, -9.7],
    [-8.0, -6.3]
])
origin = [0, 0, 0]
k_max = 100

def eta(k):
    return 0.1

def judge_function(Y, a, b):
    return 1.0 / 2 * np.sum([(np.inner(a, y) - b)**2 / np.inner(y, y)
        for y in Y])

def coff(y, a, b):
    return (b - np.inner(a, y)) / np.inner(y, y)

def batch_relaxition(s1, s2, start_a, b):
    s = [[1, x[0], x[1]] for x in s1]
    s.extend([[-1, -x[0], -x[1]] for x in s2])
    s = np.array(s)
    a = start_a
    Y = s[np.inner(a, s) <= b]
    J = [judge_function(Y, a, b)]
    k = 0
    while(len(Y) > 0 and k < k_max):
        # inp = input("iteration"+str(len(Y)))
        y = Y[0]
        # print(y,coff(y, a, b)*y)
        k += 1
        a += eta(k) * np.sum([coff(y, a, b) * y for y in Y], axis=0)
        Y = s[np.inner(a, s) <= b]
        J.append(judge_function(Y, a, b))
    return J

def single_sample_relaxition(s1, s2, start_a, b):
    s = [[1, x[0], x[1]] for x in s1]
    s.extend([[-1, -x[0], -x[1]] for x in s2])
    s = np.array(s)
    a = start_a
```

```python
    Y = s[np.inner(a, s) <= b]
    J = [judge_function(Y, a, b)]
    k = 0
    while(len(Y) > 0 and k < k_max):
        k += 1
        for y in s:
            a += eta(k) * coff(y, a, b) * y
        Y = s[np.inner(a, s) <= b]
        J.append(judge_function(Y, a, b))
    return J

if __name__ == '__main__':
    s1 = omega1
    s2 = omega3
    ax = plt.gca()
    # ax.set_xscale('log')
    J_1 = batch_relaxition(s1, s2, origin, 0.1)
    x = [i for i in range(len(J_1))]
    plt.plot(x, J_1, label="b=0.1")
    J_2 = batch_relaxition(s1, s2, origin, 0.5)
    x = [i for i in range(len(J_2))]
    plt.plot(x, J_2, label="b=0.5")
    plt.legend()
    plt.show()

    ax = plt.gca()
    # ax.set_xscale('log')
    J_1 = single_sample_relaxition(s1, s2, origin, 0.1)
    x = [i for i in range(len(J_1))]
    plt.plot(x, J_1, label="b=0.1")
    J_2 = single_sample_relaxition(s1, s2, origin, 0.5)
    x = [i for i in range(len(J_2))]
    plt.plot(x, J_2, label="b=0.5")
    plt.legend()
    plt.show()
```