University of Chinese Academy of Sciences

CS091M4042H Assignment 2 — Chapter 3

Pattern Recognition and Machine Learning

熊兴旺*
Sno: 2018E8013261007
中国科学院计算技术研究所

December 5, 2018

*email: xingw.xiong@gmail.com; xiongxingwang@ict.ac.cn

1 Problem 1

1.1 Problem Description

在一个 10 类的模式识别问题中，有 3 类单独满足多类情况 1，其余的类别满足多类情况 2。问该模式识别问题所需判别函数的最少数目是多少?

1.2 Solution

可以设计一个 2 层模式识别问题。

- 首先，把 10 类问题看成 4 类满足多类情况 1 的问题，将 3 类问题可以直接通过多类情况 1 划分得到；
- 然后，剩下 7 类问题，可以用 $\frac{7*(7-1)}{2}=21$ 个多类情况 2 的判别函数划分得到；

所以，该模式识别问题，最少需要 $4+21=25$ 个判别函数。

---

2 Problem 2

2.1 Problem Description

一个三类问题，其判别函数如下：

$$d_1(x)=-x_1, d_2(x)=x_1+x_2-1, d_3(x)=x_1-x_2-1$$

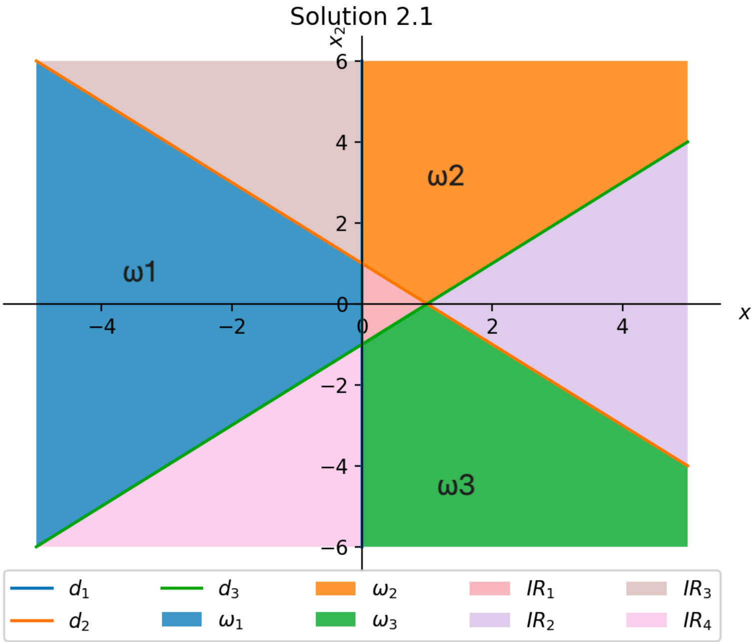
Question 1: 设这些函数是在多类情况 1 条件下确定的，绘出其判别界面和每一个模式类别的区域。

Question 2: 设为多类情况 2，并使：$d_{12}(x)=d_1(x), d_{13}(x)=d_2(x), d_{23}(x)=d_3(x)$。绘出其判别界面和多类情况 2 的区域。

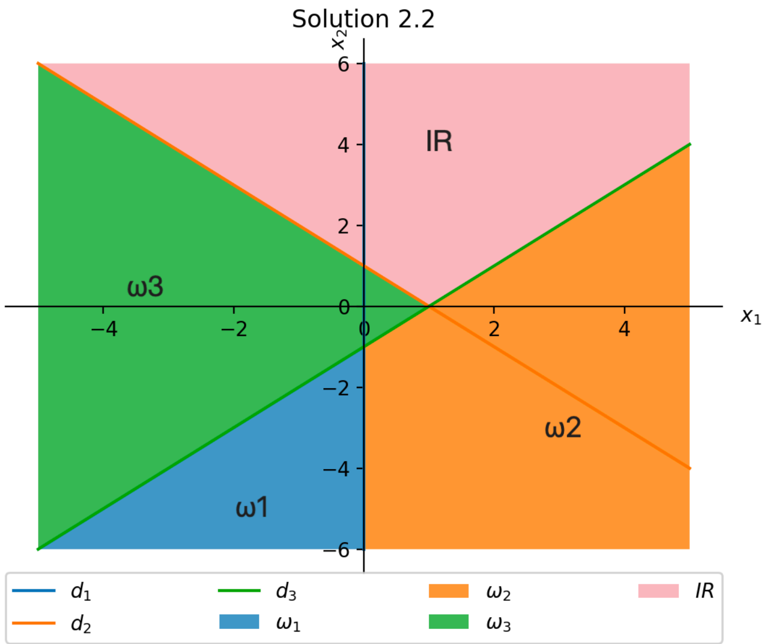
Question 3: 设 $d_1(x), d_2(x)$ 和 $d_3(x)$ 是在多类情况 3 的条件下确定的，绘出其判别界面和每类的区域。

2.2 Solution

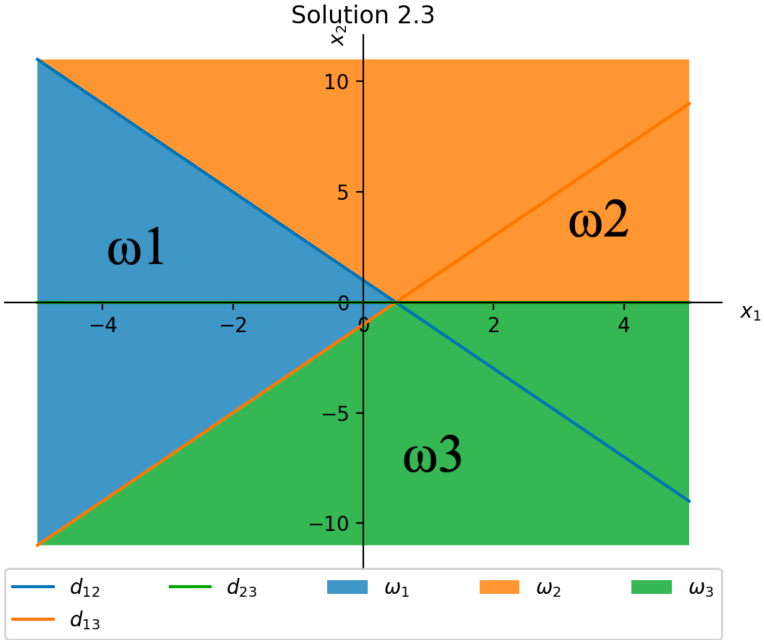
多类情况 1：判别界面和模式类别的区域绘图如下：



多类情况 2：判别界面和模式类别的区域绘图如下：



多类情况 3：判别界面和模式类别的区域绘图如下：



---

3 Problem 3

3.1 Problem Description

两类模式，每类包括 5 个 3 维不同的模式，且良好分布。如果它们是线性可分的，问权向量至少需要几个系数分量？假如要建立二次的多项式判别函数，又至少需要几个系数分量？（设模式的良好分布不因模式变化而改变。）

3.2 Solution

如果模式是线性可分的，那么判别函数可以写成：$w_0 + w_1x_1 + w_2x_2 + w_3x_3$，因此，只需要 4 个系数分量；如果是二次的多项式判别函数，那么判别函数可以写成：

$$w_0 + w_1x_1^2 + w_2x_2^3 + w_3x_3^2 + w_4x_1x_2 + w_5x_1x_3 + w_6x_2x_3 + w_7x_1 + w_8x_2 + w_9x_3 \tag{3.1}$$

因此，需要 10 个系数分量。

---

4 Problem 4

4.1 Problem Description

用感知器算法求下列模式分类的解向量 w:

$$\omega_1 : \{(0\ 0\ 0)^T, (1\ 0\ 0)^T, (1\ 0\ 1)^T, (1\ 1\ 0)^T\}$$
$$\omega_2 : \{(0\ 0\ 1)^T, (0\ 1\ 1)^T, (0\ 1\ 0)^T, (1\ 1\ 1)^T\}$$

编写求解上述问题的感知器算法程序。

4.2 Solution

下面是程序求解出来的结果:

```
Rounds <1>:
w(01)=w(00)+1*[0, 0, 0, 1]=[[0, 0, 0, 1]]
w(02)=w(01)
w(03)=w(02)
w(04)=w(03)
w(05)=w(04)+1*[0, 0, -1, -1]=[[0, 0, -1, 0]]
w(06)=w(05)
w(07)=w(06)+1*[0, -1, 0, -1]=[[0, -1, -1, -1]]
w(08)=w(07)
Rounds <2>:
w(09)=w(08)+1*[0, 0, 0, 1]=[[0, -1, -1, 0]]
w(10)=w(09)+1*[1, 0, 0, 1]=[[1, -1, -1, 1]]
w(11)=w(10)
w(12)=w(11)
w(13)=w(12)+1*[0, 0, -1, -1]=[[1, -1, -2, 0]]
w(14)=w(13)
w(15)=w(14)
w(16)=w(15)
Rounds <3>:
w(17)=w(16)+1*[0, 0, 0, 1]=[[1, -1, -2, 1]]
w(18)=w(17)
w(19)=w(18)+1*[1, 0, 1, 1]=[[2, -1, -1, 2]]
w(20)=w(19)
w(21)=w(20)+1*[0, 0, -1, -1]=[[2, -1, -2, 1]]
w(22)=w(21)
w(23)=w(22)+1*[0, -1, 0, -1]=[[2, -2, -2, 0]]
w(24)=w(23)
Rounds <4>:
w(25)=w(24)+1*[0, 0, 0, 1]=[[2, -2, -2, 1]]
w(26)=w(25)
w(27)=w(26)
w(28)=w(27)
w(29)=w(28)
w(30)=w(29)
```

```
w(31)=w(30)
w(32)=w(31)
Rounds <5>:
w(33)=w(32)
w(34)=w(33)
w(35)=w(34)
w(36)=w(35)
w(37)=w(36)
w(38)=w(37)
w(39)=w(38)
w(40)=w(39)
================
 Total Rounds:5
 Total Steps: 40
 [[ 2 -2 -2  1]]
Discriminant Function d(x): (2*x0) + (-2*x1) + (-2*x2) + 1
```

4.2.1 Code Implement

下面是感知器算法的 Python 实现：

```python
import numpy as np
X1 = np.array([(0,0,0), (1,0,0), (1,0,1), (1,1,0)])
X2 = np.array([(0,0,1), (0,1,1), (0,1,0), (1,1,1)])
X1 = np.hstack((X1, np.ones((X1.shape[0],1), dtype=np.int)))
X2 = np.hstack((X2, np.ones((X2.shape[0],1), dtype=np.int)))

X = np.vstack((X1, -X2))
w = np.zeros((X.shape[1], 1), dtype=np.int)

rounds, steps = (0, 0)
eta = 1
while True:
    rounds += 1
    finish_flag = True
    print("Rounds <%d>:" % rounds)
    for x in X:
        v = np.dot(x, w)
        steps += 1
        if v <= 0:
            w = w + eta * x.reshape(-1, 1)
            finish_flag = False
            print("w(%02d)=w(%02d)+%d*%10s=%10s" % (steps, steps-1, eta,
                str(x.tolist()), str(w.reshape(1, -1).tolist())))
        else:
            print("w(%02d)=w(%02d)" % (steps, steps-1))
    if finish_flag or rounds > 6:
        break
```

```python

print('====================\n', "Total Rounds:%d\n Total Steps: %d\n"
        % (rounds, steps), w.reshape(1, -1))

s_dx = ''
for i in range(w.shape[0]-1):
    s_dx += '(%d*x%d) + ' % (w[i], i)
s_dx += '%d' % w[-1]
print('Discriminant Function d(x): %s' % s_dx)
```

---

5 Problem 5

5.1 Problem Description

用多类感知器算法求下列模式的判别函数:

$$\omega_1: (-1\ \ -1)^T$$
$$\omega_2: (0\ \ 0)^T$$
$$\omega_3: (1\ \ 1)^T$$

5.2 Solution

将模式样本写成增广形式:

$$x_1 = (-1\ \ -1\ \ 1)^T,\ \ x_2 = (0\ \ 0\ \ 1)^T,\ \ x_3 = (1\ \ 1\ \ 1)^T$$

取初始值 $w_1(1) = w_2(1) = w_3(1) = (0\ \ 0\ \ 0)^T, C = 1$
第 1 轮迭代, 以 x_1 作为训练样本:

$$d_1(1) = w_1(1) * x_1 = 0$$
$$d_2(1) = w_2(1) * x_1 = 0$$
$$d_3(1) = w_3(1) * x_1 = 0$$

因 $d_1(1) \ngtr d_2(1)$, $d_1(1) \ngtr d_3(1)$, 故:

$$w_1(2) = w_1(1) + x_1 = (-1\ \ -1\ \ 1)^T$$
$$w_2(2) = w_2(1) - x_1 = (1\ \ 1\ \ -1)^T$$
$$w_3(2) = w_3(1) - x_1 = (1\ \ 1\ \ -1)^T$$

第 2 轮迭代, 以 x_2 作为训练样本:

$$d_1(2) = w_1(2) * x_2 = 1$$
$$d_2(2) = w_2(2) * x_2 = -1$$
$$d_3(2) = w_3(2) * x_2 = -1$$

因 $d_2(2) \ngtr d_1(2)$，$d_2(2) \ngtr d_3(2)$，故：

$$w_1(3) = w_1(1) - x_2 = (-1\ \ -1\ \ 0)^T$$
$$w_2(3) = w_2(1) + x_2 = (1\ \ 1\ \ 0)^T$$
$$w_3(3) = w_3(1) - x_2 = (1\ \ 1\ \ -2)^T$$

第 3 轮迭代，以 x_3 作为训练样本：

$$d_1(3) = w_1(3) * x_3 = -2$$
$$d_2(3) = w_2(3) * x_3 = 2$$
$$d_3(3) = w_3(3) * x_3 = 0$$

因 $d_3(3) > d_1(3)$，$d_3(3) \ngtr d_2(3)$，故：

$$w_1(4) = w_1(3) = (-1\ \ -1\ \ 0)^T$$
$$w_2(4) = w_2(3) - x_3 = (0\ \ 0\ \ -1)^T$$
$$w_3(4) = w_3(3) + x_3 = (2\ \ 2\ \ -1)^T$$

第 4 轮迭代，以 x_1 作为训练样本：

$$d_1(4) = w_1(4) * x_1 = 2$$
$$d_2(4) = w_2(4) * x_1 = -1$$
$$d_3(4) = w_3(4) * x_1 = -5$$

因 $d_1(4) > d_2(4)$，$d_1(4) > d_3(4)$，故：

$$w_1(5) = w_1(4) = (-1\ \ -1\ \ 0)^T$$
$$w_2(5) = w_2(4) = (0\ \ 0\ \ -1)^T$$
$$w_3(5) = w_3(4) = (2\ \ 2\ \ -1)^T$$

第 5 轮迭代，以 x_2 作为训练样本：

$$d_1(4) = w_1(4) * x_1 = 0$$
$$d_2(4) = w_2(4) * x_1 = -1$$
$$d_3(4) = w_3(4) * x_1 = -1$$

因 $d_2(5) \ngtr d_1(5)$，$d_2(5) \ngtr d_3(5)$，故：

$$w_1(6) = w_1(5) - x_2 = (-1\ \ -1\ \ -1)^T$$
$$w_2(6) = w_2(5) + x_2 = (0\ \ 0\ \ 0)^T$$
$$w_3(6) = w_3(5) - x_2 = (2\ \ 2\ \ -2)^T$$

第 6 轮迭代，以 x_3 作为训练样本：

$$d_1(6) = w_1(6) * x_3 = -3$$
$$d_2(6) = w_2(6) * x_3 = 0$$
$$d_3(6) = w_3(6) * x_3 = 2$$

因 $d_3(6) > d_1(6)$，$d_3(6) > d_2(6)$，故：

$$w_1(7) = w_1(6) = (-1\ \ -1\ \ -1)^T$$
$$w_2(7) = w_2(6) = (0\ 0\ 0)^T$$
$$w_3(7) = w_3(6) - x_2 = (2\ 2\ \ -2)^T$$

第 7 轮迭代，以 x_1 作为训练样本:

$$d_1(7) = w_1(7) * x_1 = 1$$
$$d_2(7) = w_2(7) * x_1 = 0$$
$$d_3(7) = w_3(7) * x_1 = -6$$

因 $d_1(7) > d_2(7)$，$d_1(7) > d_3(7)$，故：

$$w_1(8) = w_1(7) = (-1\ \ -1\ \ -1)^T$$
$$w_2(8) = w_2(7) = (0\ 0\ 0)^T$$
$$w_3(8) = w_3(7) - x_2 = (2\ 2\ \ -2)^T$$

第 8 轮迭代，以 x_2 作为训练样本:

$$d_1(8) = w_1(8) * x_2 = 1$$
$$d_2(8) = w_2(8) * x_2 = 0$$
$$d_3(8) = w_3(8) * x_2 = -2$$

因 $d_2(8) > d_1(8)$，$d_2(8) > d_3(8)$，故：

$$w_1(9) = w_1(8) = (-1\ \ -1\ \ -1)^T$$
$$w_2(9) = w_2(8) = (0\ 0\ 0)^T$$
$$w_3(9) = w_3(8) - x_2 = (2\ 2\ \ -2)^T$$

由于第 6，7，8 轮对所有样本点都已经分类正确。所以，权向量的解为:

$$w_1 = (-1\ \ -1\ \ -1)^T$$
$$w_2 = (0\ 0\ 0)^T$$
$$w_3 = (2\ 2\ \ -2)$$

3 个判别函数为:

$$d_1(\vec{x}) = [-1, -1]\,\vec{x} - 1$$
$$d_2(\vec{x}) = 0$$
$$d_3(\vec{x}) = [2, 2]\,\vec{x} - 2$$

---

6 Problem 6

6.1 Problem Description

采用梯度法和准则函数

$$J(w,x,b)=\frac{1}{8\|x\|^2}\left[\left(w^Tx-b\right)-\left|w^Tx-b\right|^2\right]$$

式中实数 $b>0$，试导出两类模式的分类算法。

6.2 Solution

对准则函数求偏导数，得:

$$\frac{\sigma J}{\sigma w}=\frac{1}{4\|x\|^2}\left[\left(w^Tx-b\right)-\left|w^Tx-b\right|\right]\left[x-x*\operatorname{sgn}\left(w^Tx-b\right)\right]$$

其中,

$$\operatorname{sgn}(x)=\begin{cases}1, & \text{if } x>0\\ -1, & \text{otherwise}\end{cases}$$

因此,

$$w(k+1)=w(k)=\begin{cases}w(k), & \text{if } w^Tx-b>0\\ w(k)-\frac{C}{\|x\|^2}\left[\left(w^Tx-b\right)x\right], & \text{otherwise}\end{cases}$$

---

7 Problem 7

7.1 Problem Description

用二次埃尔米特多项式的势函数算法求解以下模式的分类问题:

$$\omega_1:\{(0\ \ 1)^T,\ (0\ \ -1)^T\}$$
$$\omega_2:\{(1\ \ 0)^T,\ (-1\ \ 0)^T\}$$

7.2 Solution

Hermite 多项式前 3 项表达式为:

$$H_0(x)=1,\qquad H_1(x)=2x,\qquad H_2(x)=4x^2-2$$

写出二维的二次 Hermite 正交函数集:

$$
\begin{aligned}
\varphi_1(\mathbf{x}) &= \varphi_1(x_1,x_2) = H_0(x_1)H_0(x_2) = 1 \\
\varphi_2(\mathbf{x}) &= \varphi_2(x_1,x_2) = H_1(x_1)H_0(x_2) = 2x_1 \\
\varphi_3(\mathbf{x}) &= \varphi_3(x_1,x_2) = H_0(x_1)H_1(x_2) = 2x_2 \\
\varphi_4(\mathbf{x}) &= \varphi_4(x_1,x_2) = H_1(x_1)H_1(x_2) = 4x_1x_2 \\
\varphi_5(\mathbf{x}) &= \varphi_5(x_1,x_2) = H_2(x_1)H_0(x_2) = 4x_1^2-2 \\
\varphi_6(\mathbf{x}) &= \varphi_6(x_1,x_2) = H_0(x_1)H_2(x_2) = 4x_2^2-2 \\
\varphi_7(\mathbf{x}) &= \varphi_7(x_1,x_2) = H_2(x_1)H_1(x_2) = (4x_1^2-2)2x_2 \\
\varphi_8(\mathbf{x}) &= \varphi_8(x_1,x_2) = H_1(x_1)H_2(x_2) = 2x_1(4x_2^2-2) \\
\varphi_9(\mathbf{x}) &= \varphi_9(x_1,x_2) = H_2(x_1)H_2(x_2) = (4x_1^2-2)(4x_2^2-2)
\end{aligned}
$$

令:

$$\Phi(\mathbf{x}) = \begin{bmatrix}1 & 2x_1 & 2x_2 & 4x_1x_2 & 4x_1^2-2 & 4x_2^2-2 & (4x_1^2-2)2x_2 & 2x_1(4x_2^2-2) & (4x_1^2-2)(4x_2^2-2)\end{bmatrix}^T$$

因此,

$$
\begin{aligned}
\Phi(\mathbf{x}_1) &= [1 \quad 0 \quad 2 \quad 0 \quad -2 \quad 2 \quad -4 \quad 0 \quad -4]^T \\
\Phi(\mathbf{x}_2) &= [1 \quad 0 \quad -2 \quad 0 \quad -2 \quad 2 \quad 4 \quad 0 \quad -4]^T \\
\Phi(\mathbf{x}_3) &= [1 \quad 2 \quad 0 \quad 0 \quad 2 \quad -2 \quad 0 \quad -4 \quad -4]^T \\
\Phi(\mathbf{x}_4) &= [1 \quad -2 \quad 0 \quad 0 \quad 2 \quad -2 \quad 0 \quad 4 \quad -4]^T
\end{aligned}
$$

写出势函数:

$$K(\mathbf{x},\mathbf{x}_k) = \sum_{i=1}^{9}\varphi_i(x)\varphi_j(x_k) = \Phi(\mathbf{x}_k)^T\Phi(\mathbf{x})$$

通过训练样本逐步计算累积位势 $K(x)$ 的算法如下:

Step #1: 取 $\mathbf{x}_1 = (0\ 1)^T \in \omega_1$,故:

$$
\begin{aligned}
K_1(\mathbf{x}) &= K(\mathbf{x},\mathbf{x}_1) = \Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}) \\
&= [1\ 0\ 2\ 0\ -2\ 2\ -4\ 0\ -4]^T\Phi(\mathbf{x})
\end{aligned}
$$

Step #2: 取 $\mathbf{x}_2 = (0\ -1)^T \in \omega_1$,$K_1(\mathbf{x}_2) = \Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}_2) = 5 > 0$,故:

$$
\begin{aligned}
K_2(\mathbf{x}) &= K_1(\mathbf{x}) \\
&= [1\ 0\ 2\ 0\ -2\ 2\ -4\ 0\ -4]^T\Phi(\mathbf{x})
\end{aligned}
$$

Step #3: 取 $\mathbf{x}_3 = (1\ 0)^T \in \omega_2$,$K_2(\mathbf{x}_3) = \Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}_3) = 9 \nless 0$,故:

$$
\begin{aligned}
K_3(\mathbf{x}) &= K_2(\mathbf{x}) - K(\mathbf{x},\mathbf{x}_3) = \Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}) - \Phi(\mathbf{x}_3)^T\Phi(\mathbf{x}) \\
&= (\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_3))^T\Phi(\mathbf{x}) \\
&= [0\ -2\ 2\ 0\ -4\ 4\ -4\ 4\ 0]^T\Phi(\mathbf{x})
\end{aligned}
$$

Step #4: 取 $\mathbf{x}_4=(-1\ 0)^T\in\omega_2$，$K_3(\mathbf{x}_4)=(\Phi(\mathbf{x}_1)-\Phi(\mathbf{x}_3))^T\Phi(\mathbf{x}_4)=4\nless 0$，故:

$$\begin{aligned}K_4(\mathbf{x})&=K_3(\mathbf{x})-\Phi(\mathbf{x}_4)^T\Phi(\mathbf{x})\\&=(\Phi(\mathbf{x}_1)-\Phi(\mathbf{x}_3)-\Phi(\mathbf{x}_4))^T\Phi(x)\\&=[-1\ 0\ 2\ 0\ \ -6\ 6\ \ -4\ 0\ 4]^T\Phi(\mathbf{x})\end{aligned}$$

Step #5: 取 $\mathbf{x}_1=(0\ 1)^T\in\omega_1$，$K_4(\mathbf{x}_1)=27>0$，故:

$$\begin{aligned}K_5(\mathbf{x})&=K_4(\mathbf{x})\\&=[-1\ 0\ 2\ 0\ \ -6\ 6\ \ -4\ 0\ 4]^T\Phi(\mathbf{x})\end{aligned}$$

Step #6: 取 $\mathbf{x}_2=(0\ \ -1)^T\in\omega_1$，$K_5(\mathbf{x})=-13\ngtr 0$，故:

$$\begin{aligned}K_6(\mathbf{x})&=K_5(\mathbf{x})+K(\mathbf{x},\mathbf{x2})=K_5(\mathbf{x})+\Phi(\mathbf{x}_2)^T\Phi(\mathbf{x})\\&=(\Phi(\mathbf{x}_1)-\Phi(\mathbf{x}_3)+\Phi(\mathbf{x}_2)-\Phi(\mathbf{x}_4))^T\Phi(x)\\&=[0\ 0\ 0\ 0\ \ -8\ 8\ 0\ 0\ 0]^T\Phi(\mathbf{x})\end{aligned}$$

Step #7: 取 $\mathbf{x}_3=(1\ 0)^T\in\omega_2$，$K_6(\mathbf{x}_3)=-32<0$，故:

$$\begin{aligned}K_7(\mathbf{x})&=K_6(\mathbf{x})\\&=(\Phi(\mathbf{x}_1)-\Phi(\mathbf{x}_3)+\Phi(\mathbf{x}_2)-\Phi(\mathbf{x}_4))^T\Phi(x)\\&=[0\ 0\ 0\ 0\ \ -8\ 8\ 0\ 0\ 0]^T\Phi(\mathbf{x})\end{aligned}$$

Step #8: 取 $\mathbf{x}_4=(-1\ 0)^T\in\omega_2<0$，$K_7(\mathbf{x}_4)=-32<0$，故:

$$\begin{aligned}K_8(\mathbf{x})&=K_7(\mathbf{x})\\&=(\Phi(\mathbf{x}_1)-\Phi(\mathbf{x}_3)+\Phi(\mathbf{x}_2)-\Phi(\mathbf{x}_4))^T\Phi(x)\\&=[0\ 0\ 0\ 0\ \ -8\ 8\ 0\ 0\ 0]^T\Phi(\mathbf{x})\end{aligned}$$

Step #9: 取 $\mathbf{x}_1=(0\ 1)^T\in\omega_1$，$K_8(\mathbf{x}_1)=32>0$，故:

$$\begin{aligned}K_9(\mathbf{x})&=K_8(\mathbf{x})\\&=(\Phi(\mathbf{x}_1)-\Phi(\mathbf{x}_3)+\Phi(\mathbf{x}_2)-\Phi(\mathbf{x}_4))^T\Phi(x)\\&=[0\ 0\ 0\ 0\ \ -8\ 8\ 0\ 0\ 0]^T\Phi(\mathbf{x})\end{aligned}$$

Step #10: 取 $\mathbf{x}_2=(0\ \ -1)^T\in\omega_1$，$K_9(\mathbf{x}_2)=32>0$，故:

$$\begin{aligned}K_{10}(\mathbf{x})&=K_9(\mathbf{x})\\&=(\Phi(\mathbf{x}_1)-\Phi(\mathbf{x}_3)+\Phi(\mathbf{x}_2)-\Phi(\mathbf{x}_4))^T\Phi(x)\\&=[0\ 0\ 0\ 0\ \ -8\ 8\ 0\ 0\ 0]^T\Phi(\mathbf{x})\end{aligned}$$

以上对全部训练样本都能正确分类，因此算法收敛于判别函数:

$$\begin{aligned}d(\mathbf{x})&=[0\ 0\ 0\ 0\ \ -8\ 8\ 0\ 0\ 0]^T\Phi(\mathbf{x})\\&=-8(4x_1^2-2)+8(4x_2^2-2)\\&=-32(x_1^2-x_2^2)\end{aligned}$$

下面是 Python 程序实现代码:

```python
import numpy as np

X1=np.array([(0, 1), (0, -1)])
X2=np.array([(1, 0), (-1, 0)])
Y1=np.full((X1.shape[0], 1), +1, dtype=np.int)
Y2=np.full((X2.shape[0], 1), -1, dtype=np.int)
X =np.vstack((X1, X2))
Y =np.vstack((Y1, Y2))

def Phi(x):
    x1, x2=x
    return np.array([1, 2*x1, 2*x2,
        4*x1*x2, 4*x1*x1-2, 4*x2*x2-2,
        (4*x1*x1-2)*2*x2, 2*x1*(4*x2*x2-2),
        (4*x1*x1-2)*(4*x2*x2-2)])

rounds = 0
steps = 0
match_cnt = 0
K = [np.zeros(9, dtype=np.int), ]

while match_cnt < X.shape[0]:
    rounds += 1
    for x,y in zip(X,Y):
        if match_cnt >= X.shape[0]: break
        steps += 1
        Kt = Phi(x)
        v = np.dot(K[-1], Kt)
        if v*y <= 0:
            K.append(K[-1] + y * Kt)
            match_cnt = 0
        else:
            match_cnt += 1
        print('Step#%d: K(x%d)=%d\n' % (steps, (steps+3)%4+1, v), K[-1], '\n')
```

---

8 Problem 8

8.1 Problem Description

用下列势函数:

$$K(x, x_k) = e^{-\alpha\|x-x_k\|^2}$$

求解以下模式的分类问题:

$$\omega_1:\{(0\ 1)^T,\ (0\ -1)^T\}$$
$$\omega_2:\{(1\ 0)^T,\ (-1\ 0)^T\}$$

8.2 Solution

势函数 $K(\mathbf{x},\mathbf{x}_k)=e^{-\alpha\|\mathbf{x}-\mathbf{x}_k\|^2}$，这里，取 $\alpha=1$，那么，势函数模式分类的步骤如下：

Step #1: 取 $\mathbf{x}_1=(0\ 1)^T\in\omega_1$，故:

$$\begin{aligned}K_1(\mathbf{x})&=K(\mathbf{x},\mathbf{x}_1)=e^{-\|\mathbf{x}-\mathbf{x}_1\|^2}\\&=e^{x_1^2+(x_2-1)^2}\end{aligned}$$

Step #2: 取 $\mathbf{x}_2=(0\ -1)^T\in\omega_1$， $K_1(\mathbf{x}_2)=0.018316>0$，故:

$$\begin{aligned}K_2(\mathbf{x})&=K_1(\mathbf{x})\\&=e^{x_1^2+(x_2-1)^2}\end{aligned}$$

Step #3: 取 $\mathbf{x}_3=(1\ 0)^T\in\omega_2$， $K_2(\mathbf{x}_3)=0.135335\nless 0$，故:

$$\begin{aligned}K_3(\mathbf{x})&=K_2(\mathbf{x})-K(\mathbf{x},\mathbf{x}_3)\\&=e^{x_1^2+(x_2-1)^2}-e^{(x_1-1)^2+x_2^2}\end{aligned}$$

Step #4: 取 $\mathbf{x}_4=(-1\ 0)^T\in\omega_2$， $K_3(\mathbf{x}_4)=0.117020\nless 0$，故

$$\begin{aligned}K_4(\mathbf{x})&=K_3(\mathbf{x})-K(\mathbf{x},\mathbf{x}_4)\\&=e^{x_1^2+(x_2-1)^2}-e^{(x_1-1)^2+x_2^2}-e^{(x_1+1)^2+x_2^2}\end{aligned}$$

Step #5: 取 $\mathbf{x}_1=(0\ 1)^T\in\omega_1$， $K_4(\mathbf{x}_1)=0.729329>0$，故:

$$\begin{aligned}K_5(\mathbf{x})&=K_4(\mathbf{x})\\&=e^{x_1^2+(x_2-1)^2}-e^{(x_1-1)^2+x_2^2}-e^{(x_1+1)^2+x_2^2}\end{aligned}$$

Step #6: 取 $\mathbf{x}_2=(0\ -1)^T\in\omega_1$， $K_5(\mathbf{x}_2)=-0.252355\ngtr 0$，故:

$$\begin{aligned}K_6(\mathbf{x})&=K_5(\mathbf{x})+K(\mathbf{x},\mathbf{x}_2)\\&=e^{x_1^2+(x_2-1)^2}-e^{(x_1-1)^2+x_2^2}-e^{(x_1+1)^2+x_2^2}+e^{x_1^2+(x_2+1)^2}\end{aligned}$$

Step #7: 取 $\mathbf{x}_3=(1\ 0)^T\in\omega_2$， $K_6(\mathbf{x}_3)=-0.747645<0$，故:

$$\begin{aligned}K_7(\mathbf{x})&=K_6(\mathbf{x})\\&=e^{x_1^2+(x_2-1)^2}-e^{(x_1-1)^2+x_2^2}-e^{(x_1+1)^2+x_2^2}+e^{x_1^2+(x_2+1)^2}\end{aligned}$$

Step #8: 取 $\mathbf{x}_4=(-1\ 0)^T\in\omega_2$， $K_7(\mathbf{x}_4)=-0.747645<0$，故:

$$\begin{aligned}K_8(\mathbf{x})&=K_7(\mathbf{x})\\&=e^{x_1^2+(x_2-1)^2}-e^{(x_1-1)^2+x_2^2}-e^{(x_1+1)^2+x_2^2}+e^{x_1^2+(x_2+1)^2}\end{aligned}$$

Step #9: 取 $\mathbf{x}_1=(0\ \ 1)^T\in\omega_1$， $K_8(\mathbf{x}_1)=0.747645>0$， 故:

$$\begin{aligned}K_9(\mathbf{x}) &= K_8(\mathbf{x})\\ &= e^{x_1^2+(x_2-1)^2}-e^{(x_1-1)^2+x_2^2}-e^{(x_1+1)^2+x_2^2}+e^{x_1^2+(x_2+1)^2}\end{aligned}$$

Step #10: 取 $\mathbf{x}_2=(0\ \ -1)^T\in\omega_1$， $K_9(\mathbf{x}_2)=0.747645>0$， 故:

$$\begin{aligned}K_{10}(\mathbf{x}) &= K_9(\mathbf{x})\\ &= e^{x_1^2+(x_2-1)^2}-e^{(x_1-1)^2+x_2^2}-e^{(x_1+1)^2+x_2^2}+e^{x_1^2+(x_2+1)^2}\end{aligned}$$

经过上述迭代，全部模式都已正确分类，因此算法收敛于判别函数:

$$d(\mathbf{x})=e^{x_1^2+(x_2-1)^2}-e^{(x_1-1)^2+x_2^2}-e^{(x_1+1)^2+x_2^2}+e^{x_1^2+(x_2+1)^2}$$

下面是 Python 程序实现代码:

```python
import numpy as np

X1=np.array([(0,1), (0,-1)])
X2=np.array([(1,0), (-1,0)])
Y1=np.full((X1.shape[0], 1), +1, dtype=np.int)
Y2=np.full((X2.shape[0], 1), -1, dtype=np.int)
X =np.vstack((X1, X2))
Y =np.vstack((Y1, Y2))

rounds, steps, match_cnt = (0, 0, 0)
params = []

while match_cnt < X.shape[0] and rounds < 200:
    rounds += 1
    for x, y in zip(X, Y):
        if match_cnt >= X.shape[0]: break
        steps += 1
        v = sum([py*np.exp(-pow(np.linalg.norm(px-x, ord=2), 2)) for px,py in params])
        #v = sum([py*np.exp(-np.square(px-x).sum()) for (px,py) in params])
        if v * y <= 0:
            params.append((x, y))
            match_cnt = 0
        else:
            match_cnt += 1
        print('Step#%d: K(x%d)=%f\n' % (steps, (steps+3)%4+1, v),
              [(x.tolist(), y.tolist()) for x,y in params], '\n')
```