1. $X \in R^3$, 3D perceptron: $h_w(x) = \text{sign}\left(\sum_{i=0}^{3} w_i x_i\right)$

(b)
$$\begin{bmatrix} x_3^1 & x_2^1 & x_1^1 & 1 \\ x_3^2 & x_2^2 & x_1^2 & 1 \\ & \vdots & & \\ x_3^n & x_2^n & x_1^n & 1 \end{bmatrix} \begin{bmatrix} w_3 \\ w_2 \\ w_1 \\ w_0 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$X \cdot W = Y$$

range $(X) = R^n \Rightarrow$ all X can be shatter

(a) $\text{rank}\left(\begin{bmatrix} 2 & 3 & 4 & 1 \\ 4 & 3 & 2 & 1 \\ 3 & 3 & 3 & 1 \end{bmatrix}\right) = 2 < 3$

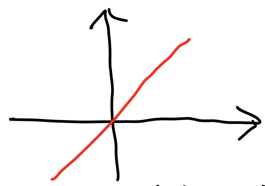
(b) $\text{rank}\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 4 & 1 \\ 4 & 3 & 2 & 1 \\ 4 & 2 & 3 & 1 \end{bmatrix}\right) = 4$

(c) $\text{rank}\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 4 & 1 \\ 4 & 3 & 2 & 1 \\ 2 & 2 & 2 & 1 \end{bmatrix}\right) = 3 < 4$

(d) $\text{rank}\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 4 & 1 \\ 4 & 3 & 2 & 1 \\ 4 & 2 & 3 & 1 \\ 3 & 2 & 4 & 1 \end{bmatrix}\right) = 4 < 5$

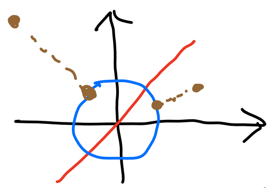
2, $W_0 = 0$ perceptron cross original point

(c)

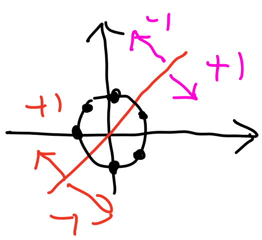


if two points with the same polar angle, they will be classfied with the same label

so we can transform R^2 into unit circle
(map all points to the unit circle with same angle)

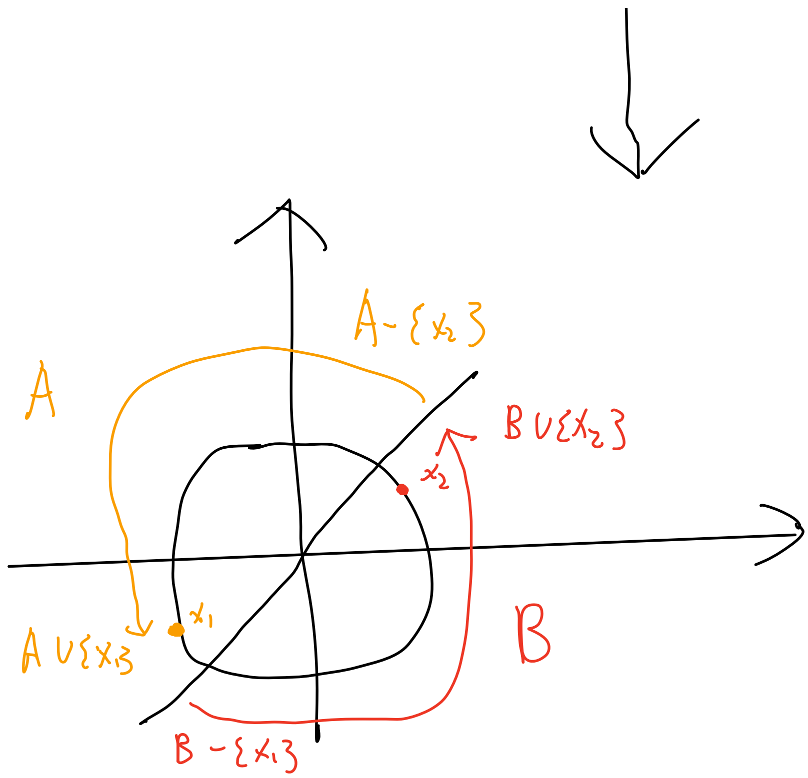


perceptron can seperate n point into two group
these two group can be +1/-1 or -1/+1



at most n combination

2n dichotomy in total



rotate at most 180° until scan all point

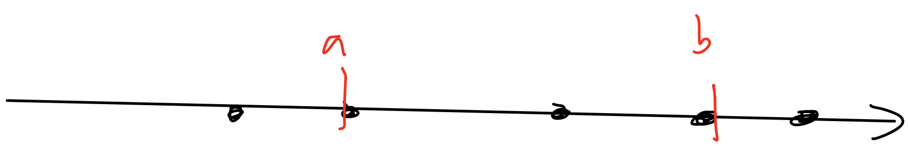
⇒ at most n 2 group combination

3 $\sum_{i=1}^{d} x_i^2 = (\text{eucidean distance between } x \text{ and original point})^2$

(a) all x which eucidean distance to original point is same will have same label.

$\Rightarrow$ map $R^d \to R^1$

$$\begin{bmatrix} x_d \\ \vdots \\ x_1 \end{bmatrix} \mapsto \sum_{i=1}^{d} x_i^2$$



Same as positive interval

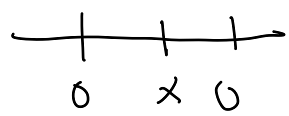
$$\binom{n+1}{2} + 1$$

4

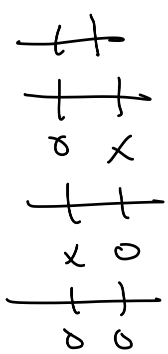
followed question 3

(d)

we cannot shatter these label from all input



$N=3$



$N=2$

we can shatter 2 input

$VC = 2$

5

(a) positive interval 1

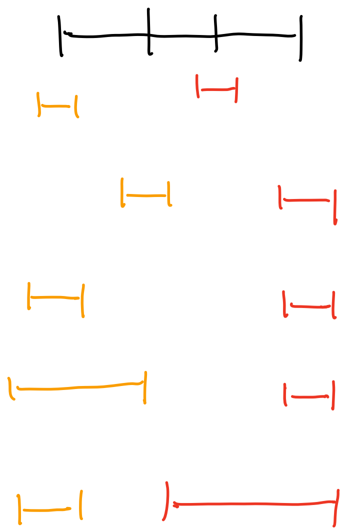
positive interval 2

(e) $n = 5$:

o x o x o

we can not shatter this dichonomy because we only have 2 interval

$n = 4 \Rightarrow$ we can shatter all dichonomy



o x o x

x o x o

o x x o

o o x o

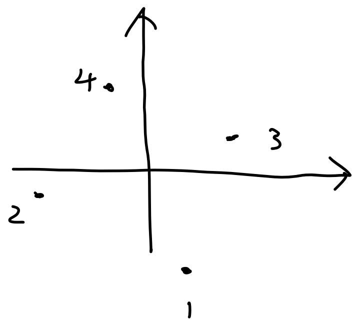
o x o o

these dichonomy can't be shatter by 1 interval $\Leftarrow$ can be shatter by 2 interval

dichonomy which can be shatter by 1 interval can also be shatter by 2 interval

$d_{vc} = 4$

(b) $n=4$



These input can be shatter by classifier

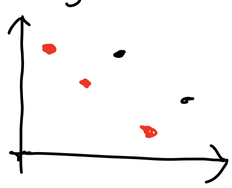
$n=5$

let $(x_1, y_1), (x_2, y_2), (x_3, y_3)\ (x_4, y_4)\ (x_5, y_5)$

without loss of generality let $x_1 < x_2 < x_3 < x_4 < x_5$

y may have $5!$ order

in y there exist at least 3 is in decrease / increase order ex



$\Rightarrow$ these 3 point can not shatter o x o sub-dichonomy

$\Rightarrow$ 5 point can not be shatter

$d_{vc} = 4$

(c) $n = 4$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}}_{\text{range} = R^4 = Y} \begin{bmatrix} w_4 \\ w_3 \\ w_2 \\ w_1 \\ w_0 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

These x can be shatter

$n = 5$

$$\begin{bmatrix} x_4^1 & x_3^1 & x_2^1 & x_1^1 & 1 \\ x_4^2 & x_3^2 & x_2^2 & x_1^2 & 1 \\ x_4^3 & x_3^3 & x_2^3 & x_1^3 & 1 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 & 1 \\ x_4^5 & x_3^5 & x_2^5 & x_1^5 & 1 \end{bmatrix} \begin{bmatrix} w_5 \\ w_4 \\ w_3 \\ w_2 \\ w_1 \\ w_0 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

$$w_0 = \frac{\begin{vmatrix} x_4^1 & x_3^1 & x_2^1 & x_1^1 & y_1 \\ x_4^2 & x_3^2 & x_2^2 & x_1^2 & y_2 \\ x_4^3 & x_3^3 & x_2^3 & x_1^3 & y_3 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 & y_4 \\ x_4^5 & x_3^5 & x_2^5 & x_1^5 & y_5 \end{vmatrix}}{\begin{vmatrix} x_4^1 & x_3^1 & x_2^1 & x_1^1 & 1 \\ x_4^2 & x_3^2 & x_2^2 & x_1^2 & 1 \\ x_4^3 & x_3^3 & x_2^3 & x_1^3 & 1 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 & 1 \\ x_4^5 & x_3^5 & x_2^5 & x_1^5 & 1 \end{vmatrix}}$$

By Cramer's rule

$$
\frac{y_1 \overset{A}{\begin{vmatrix} x_4^2 & x_3^2 & x_2^2 & x_1^2 \\ x_4^3 & x_3^3 & x_2^3 & x_1^3 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 \\ x_4^5 & x_3^5 & x_2^5 & x_1^5 \end{vmatrix}} - y_2 \overset{B}{\begin{vmatrix} x_4^1 & x_3^1 & x_2^1 & x_1^1 \\ x_4^3 & x_3^3 & x_2^3 & x_1^3 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 \\ x_4^5 & x_3^5 & x_2^5 & x_1^5 \end{vmatrix}} + y_3 \overset{C}{\begin{vmatrix} x_4^1 & x_3^1 & x_2^1 & x_1^1 \\ x_4^2 & x_3^2 & x_2^2 & x_1^2 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 \\ x_4^5 & x_3^5 & x_2^5 & x_1^5 \end{vmatrix}} - y_4 \overset{D}{\begin{vmatrix} x_4^1 & x_3^1 & x_2^1 & x_1^1 \\ x_4^2 & x_3^2 & x_2^2 & x_1^2 \\ x_4^3 & x_3^3 & x_2^3 & x_1^3 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 \end{vmatrix}}}{\underset{A}{\begin{vmatrix} x_4^2 & x_3^2 & x_2^2 & x_1^2 \\ x_4^3 & x_3^3 & x_2^3 & x_1^3 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 \\ x_4^5 & x_3^5 & x_2^5 & x_1^5 \end{vmatrix}} - \underset{B}{\begin{vmatrix} x_4^1 & x_3^1 & x_2^1 & x_1^1 \\ x_4^3 & x_3^3 & x_2^3 & x_1^3 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 \\ x_4^5 & x_3^5 & x_2^5 & x_1^5 \end{vmatrix}} + \underset{C}{\begin{vmatrix} x_4^1 & x_3^1 & x_2^1 & x_1^1 \\ x_4^2 & x_3^2 & x_2^2 & x_1^2 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 \\ x_4^5 & x_3^5 & x_2^5 & x_1^5 \end{vmatrix}} - \underset{D}{\begin{vmatrix} x_4^1 & x_3^1 & x_2^1 & x_1^1 \\ x_4^2 & x_3^2 & x_2^2 & x_1^2 \\ x_4^3 & x_3^3 & x_2^3 & x_1^3 \\ x_4^4 & x_3^4 & x_2^4 & x_1^4 \end{vmatrix}}}
$$

$$> 0$$

without loss of generality

let $A - B + C - D > 0$

$$A + C > B + D$$

let $y_4 = \text{sign}(D)$ $\quad y_2 = \text{sign}(B)$

$y_1 = -\text{sign}(A)$ $\quad y_3 = -\text{sign}(C)$

$\Rightarrow$ $w_0 < 0$ contradict

so we can not shatter any 5 point

$$d_{VC} = 4$$

(e) $n=4$

$$\begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ & \vdots & & \\ x_n^3 & x_n^2 & x_n & 1 \end{bmatrix} \begin{bmatrix} w3 \\ w2 \\ w1 \\ w0 \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

take an example $\begin{bmatrix} 1^3 & 1^2 & 1 & 1 \\ 2^3 & 2^2 & 2 & 1 \\ 3^3 & 3^2 & 3 & 1 \\ 0^3 & 0^2 & 0 & 1 \end{bmatrix}$

column independent $\Rightarrow$ invertable

$\Rightarrow$ Range $(X) \in R^4 \Rightarrow$ can be shatter

$n=5$

$$\begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ x_3^3 & x_3^2 & x_3 & 1 \\ x_4^3 & x_4^2 & x_4 & 1 \\ x_5^3 & x_5^2 & x_5 & 1 \end{bmatrix}$$

$x5 = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4$

$$\underbrace{w^T x_5} = a_1 \underbrace{w^T x_1} + a_2 \underbrace{w^T x_2} + a_3 \underbrace{w^T x_3} + a_4 \underbrace{w^T x_4}$$

for all x we can not generate dichotomy

$(\text{sign}(a_1), \text{sign}(a_2), \text{sign}(a_3), \text{sign}(a_4), -1)$

$d_{vc} = 4$

6 $d_{vc}(H) = N \Rightarrow$ We can shatter at most N point

$\Rightarrow$ There exist 2^N dichotomies

$\Rightarrow$ at least 2^N classifiers

(d)

case 1:

$d_{vc}(H) = 11 \Rightarrow$ at least 2048 classifiers

case 2:

$d_{vc}(H) = 10 \Rightarrow$ at least 1024 classifiers

maximum $d_{vc}(H) = 10$

η

(d) revise hoeffding inequality

origin:

$$P(|E_{in}(h) - E_{out}(h)| > \epsilon) \leq \underbrace{2M\exp(-2\epsilon^2 N)}_{\delta}$$

$$\delta = 2M\exp(-2\epsilon^2 N)$$

$$\frac{\delta}{2M} = \exp(-2\epsilon^2 N)$$

$$\ln\left(\frac{2M}{\delta}\right) = 2\epsilon^2 N$$

$$\sqrt{\frac{1}{2N}\ln\left(\frac{2M}{\delta}\right)} = \epsilon$$

probability $> 1-\delta$

$$|E_{in}(h) - E_{out}(h)| \leq \sqrt{\frac{1}{2N}\ln\left(\frac{2M}{\delta}\right)}$$

$$\Rightarrow E_{in}(h) - \sqrt{\frac{1}{2N}\ln\left(\frac{2M}{\delta}\right)} \leq E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{1}{2N}\ln\left(\frac{2M}{\delta}\right)}$$

① $E_{out}(g) \leq E_{in}(g) + \epsilon$

$$E_{out}(g) - E_{out}(g^*) \leq E_{in}(g) - E_{out}(g^*) + \epsilon$$

② $E_{in}(g^*) - \epsilon \le E_{out}(g^*)$

$$\underbrace{E_{in}(g^*) - E_{in}(g)}_{\ge 0} - \epsilon \le E_{out}(g^*) - E_{in}(g)$$

$$\epsilon \ge \epsilon - \underbrace{E_{in}(g^*) - E_{in}(g)}_{\ge 0} \ge E_{in}(g) - E_{out}(g^*)$$

combine ①、②

$$E_{out}(g) - E_{out}(g^*) \le \epsilon + \epsilon = 2\sqrt{\frac{1}{2N}\ln\left(\frac{2M}{\delta}\right)}$$

8

(b) VC bound:

$$P[\exists h \in H \text{ s.t. } |E_{in}(h) - E_{out}(h)| > \epsilon] \le 4 m_H(2N) \exp\left(-\frac{\epsilon^2 N}{8}\right)$$

positive ray: $m_H(2N) = 2N+1$ $\quad \epsilon = 0.1 \quad \delta = 0.1$

$$P[\exists h \in H \text{ st } |E_{in}(h) - E_{out}(h)| > 0.1] \le (8N+4)\exp\left(\frac{-0.01N}{8}\right)$$

$$(8N+4)\exp\left(\frac{-N}{800}\right) \le 0.1$$

(a) 0.2981

(b) 0.043

(c) 0.029

(d) 0.009

(e) 0.002

a

$$W = \underbrace{u}_{\text{known}} + \underbrace{v}_{\text{unknown}}$$

$$\min E(u+v) \approx \min E(u) + b_E(u)^T v + \frac{1}{2} v^T A_E(u) v$$

By FONC

(b)

$$0 = b_E(u) + A_E(u) v$$

$$v = - A_E(u)^{-1} b_E(u)$$

10 let x_{ij} = j-th dimention in ith vector

(d) (i, j) element in $A_E(w)$ is $\frac{\partial^2 E}{\partial w_i \partial w_j}(w)$

$$\frac{\partial E}{\partial w_i}(w) = \frac{1}{N}\sum_{n=1}^{N}\left[\frac{1}{1+\exp(-y_n w^T x_n)}\exp(-y_n w^T x_n)(-y_n x_{ni})\right]$$

$$\Downarrow$$

$$\frac{\partial E}{\partial w_i w_j}(w) = \frac{1}{N}\sum_{n=1}^{N}\left[\frac{\partial\left(\frac{\exp(-y_n w^T x_n)}{1+\exp(-y_n w^T x_n)}\right)}{\partial w_j}(-y_n x_{ni}) + \cancel{\frac{\exp(-y_n w^T x_n)}{1+\exp(-y_n w^T x_n)}}\underbrace{\frac{\partial(-y_n x_{ni})}{\partial w_j}}_{0}\right]$$

$$\frac{\partial\left(\frac{\exp(-y_n w^T x_n)}{1+\exp(-y_n w^T x_n)}\right)}{\partial w_j} = \frac{\partial\left(\frac{1}{\frac{1}{\exp(-y_n w^T x_n)}+1}\right)}{\partial w_j} =$$

$$\underbrace{\left(\frac{1}{\frac{1}{\exp(-y_n w^T x_n)}+1}\right)^2\left(\frac{1}{\exp(-y_n w^T x_n)}\right)^2\exp(-y_n w^T x_n)}_{\underbrace{\frac{1}{1+\exp(-y_n w^T x_n)}}_{h_t(-y_n x_n)}\ \underbrace{\frac{1}{1+\frac{1}{\exp(-y_n w^T x_n)}}}_{\frac{1}{1+\exp(y_n w^T x_n)} = h_t(y_n x_n)}}\underbrace{(-y_n x_{nj})(-y_n x_{ni})}_{x_{ni}\ x_{nj}}$$

$$\Rightarrow A_E(w_t) = \frac{1}{N}\sum_{n=1}^{N} h_t(y_n x_n)\, h_t(-y_n x_n)\, x_n x_n^T$$

11

1(c) XX^T is a N by N matrix which it's diagonal side is just like Σ matrix but Σ singular value change to 1

number of singular value

$$XX^T = \begin{bmatrix} 1 & & & 0 & \\ & 1 & & & \\ & & 1 & & \\ & 0 & & 1 & \\ & & & & \ddots \\ & & & & & 0 \end{bmatrix}_{N \times N}$$

12

(a)

$$\text{likelihood} = \underbrace{p(x_1)}_{\text{fixed}} \Big(\underbrace{\frac{1}{a\sqrt{2\pi}}}_{\text{fixed}} e^{-\frac{1}{2}\left(\frac{y_1 - w x_1}{a}\right)} \Big)$$

$$\underbrace{p(x_2)}_{\text{fixed}} \Big(\underbrace{\frac{1}{a\sqrt{2\pi}}}_{\text{fixed}} e^{-\frac{1}{2}\left(\frac{y_2 - w^T x_2}{a}\right)^2} \Big)$$

$$\vdots$$

$$\underbrace{p(x_n)}_{\text{fixed}} \Big(\underbrace{\frac{1}{a\sqrt{2\pi}}}_{\text{fixed}} e^{-\frac{1}{2}\left(\frac{y_n - w^T x_n}{a}\right)^2} \Big)$$

$$\text{likelihood} \propto \ln\left(e^{-\frac{1}{2}\left(\frac{y_1 - w^T x_1}{a}\right)^2} \cdots e^{-\frac{1}{2}\left(\frac{y_n - w^T x_n}{a}\right)^2} \right)$$

$$\propto -\left((y_1 - w^T x_1)^2 + \cdots (y_n - w^T x_n)^2 \right)$$

maximize liklihood $\Rightarrow$ minimize $(y_1 - w^T x_1)^2 + \cdots (y_n - w^T x_n)^2$

$$= -2 w^T X^T Y + w^T X^T X w$$

quadratic function

$X^T X$ is semi-positive-definite

$\Rightarrow$ convex function

$\Rightarrow$ min in $\nabla E(x) = 0$

$$-2X^T y + 2X^T X w = 0$$

$$\Rightarrow \quad X^T X w = X^T y$$

$$\Rightarrow \quad w = (X^T X)^{-1} X^T y$$

13 (a) 0.037

14 (e) 0.0034

15 (b) (0.057, 0.0589)

16 (c) (0.0404, 0.0584)

```python
import numpy as np
import random
import math
def train_generate(x_data, y_data):
    for i in range(200):
        y = random.choice([-1, 1])
        if y == 1:                              # mean = [2, 3]  cov = [[0.6, 0], [0, 0.6]]
            x1 = random.normalvariate(2, math.sqrt(0.6))
            x2 = random.normalvariate(3, math.sqrt(0.6))
            x_data.append([1.0, x1, x2])
            y_data.append([1.0])
        else:                                   # mean = [0, 4]  cov = [[0.4, 0], [0, 0.4]]
            x1 = random.normalvariate(0, math.sqrt(0.4))
            x2 = random.normalvariate(4, math.sqrt(0.4))
            x_data.append([1.0, x1, x2])
            y_data.append([-1.0])

    for i in range(20):                          #outlier data
        x1 = random.normalvariate(6, math.sqrt(0.3))
        x2 = random.normalvariate(0, math.sqrt(0.1))
        x_data.append([1.0, x1, x2])
        y_data.append([1.0])


def test_generate(x_data, y_data):
    for i in range(5000):
        y = random.choice([-1, 1])
        if y == 1:                              # mean = [2, 3]  cov = [[0.6, 0], [0, 0.6]]
            x1 = random.normalvariate(2, math.sqrt(0.6))
            x2 = random.normalvariate(3, math.sqrt(0.6))
            x_data.append([1.0, x1, x2])
            y_data.append([1.0])
        else:                                   # mean = [0, 4]  cov = [[0.4, 0], [0, 0.4]]
            x1 = random.normalvariate(0, math.sqrt(0.4))
            x2 = random.normalvariate(4, math.sqrt(0.4))
            x_data.append([1.0, x1, x2])
            y_data.append([-1.0])

def sigmoid(x):
    return 1/(1+np.exp(-1*x))

def main():
    iter = 100
    e_in = 0
    e_out = 0
    train_size = 200
    test_size = 5000


    for i in range(iter):
        random.seed(i)
        x_traindata = []
        y_traindata = []
        train_generate(x_traindata, y_traindata)
        x_traindata = np.array(x_traindata)
        y_traindata = np.array(y_traindata)

        w_op = np.linalg.pinv(x_traindata).dot(y_traindata)
        predict_trainy = x_traindata.dot(w_op)

        w_op = np.array([[0.0],[0.0],[0.0]])
        lr = 0.1
        T = 500
        for j in range(T):
            gd = np.array([[0.0],[0.0],[0.0]])
            for k in range(train_size):
```

```
            gd+=sigmoid(-1*y_traindata[k, 0]*w_op.T.dot(x_traindata[k,]))*y_traindata[k, 0]*x_traindata[k,].reshape(3,1)
        gd/=train_size
        w_op+=lr*gd

      #e_in+=((np.sign(predict_trainy)!=y_traindata).sum()/train_size)
      #e_in+=np.linalg.norm(predict_trainy-y_traindata)/train_size

    ##################################################################

      x_testdata = []
      y_testdata = []
      test_generate(x_testdata, y_testdata)
      x_testdata = np.array(x_testdata)
      y_testdata = np.array(y_testdata)

      predict_testy = x_testdata.dot(w_op)

      e_out+=((np.sign(predict_testy)!=y_testdata).sum()/test_size)
      #e_out+=((np.sign(predict_testy)!=y_testdata).sum()/test_size)
      #e_out+=np.linalg.norm(predict_testy-y_testdata)/test_size

    e_in/=iter
    e_out/=iter
    print(e_in, e_out, abs(e_in-e_out))

if __name__ == '__main__':
    main()
```