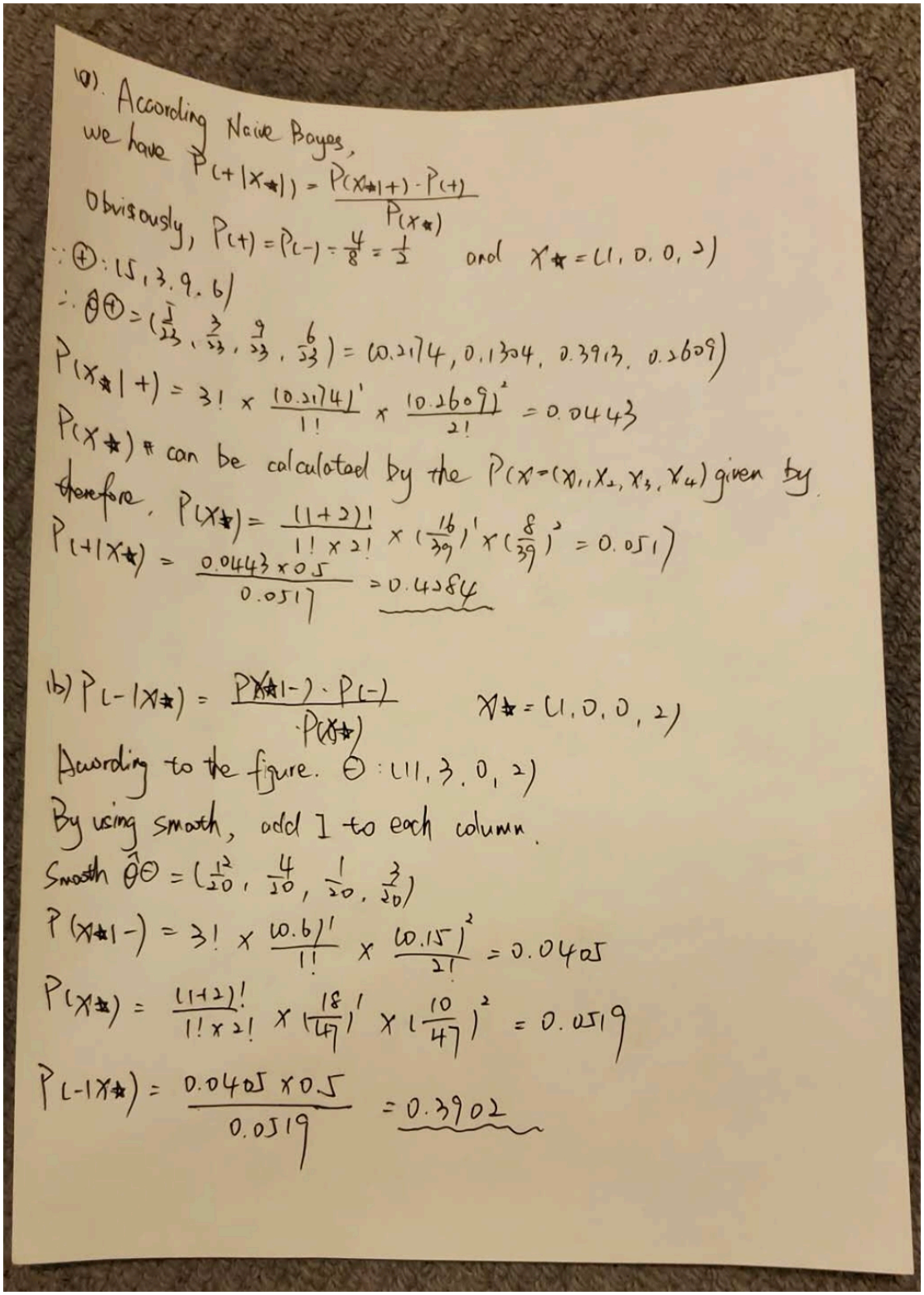
zID：z5224151
Name: ZANNING WANG

Q1:



(c) Similar as last two questions, but we should translate x to bit vector: $(1,0,0,1)$

$$P(+|x_*) = \frac{P(x|+)\cdot P(+)}{P(x_*)} \qquad P(+) = P(-) = \frac{4}{8} = \frac{1}{2}$$

$$P(x_*|+) = \frac{1}{2} \times (1-\frac{1}{4}) \times (1-\frac{3}{4}) \times \frac{1}{2} = 0.0468$$

$$P(x_*) = \frac{3}{8} \times (1-\frac{2}{8}) \times (1-\frac{3}{8}) \times \frac{4}{8} = 0.1464$$

$$P(+|x_*) = \frac{0.0468 \times 0.5}{0.1464} = \underline{0.1598}$$

(d) $P(x|-) = \frac{4}{6} \times (1-\frac{2}{6}) \times (1-\frac{1}{6}) \times \frac{3}{6} = 0.1851$

In order to smooth, we should add $(0,0,0)$ and $(1,1,1)$ to both $\oplus$, $\ominus$ classes,

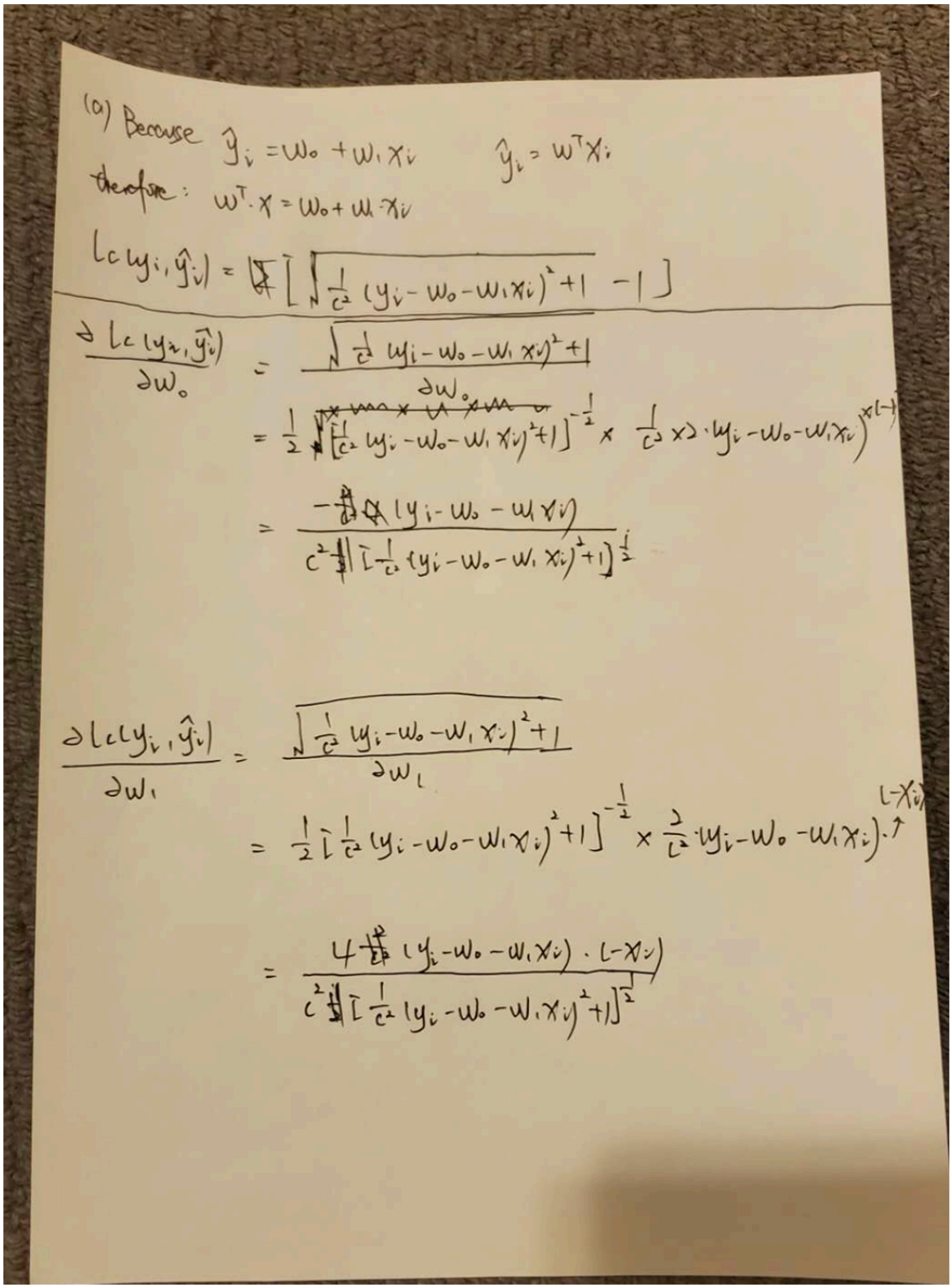
~~therefore~~:

$$P(x_*) = \frac{7}{12} \times (1-\frac{4}{12}) \times (1-\frac{4}{12}) \times \frac{6}{12} = 0.1296$$

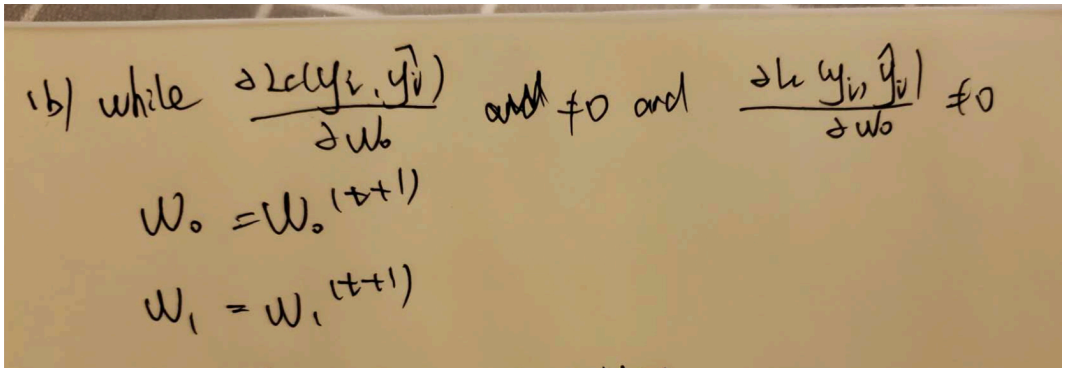
$$P(-|x_*) = \frac{0.1851 \times 0.5}{0.1296} = \underline{0.7141}$$

Q2:

(a)

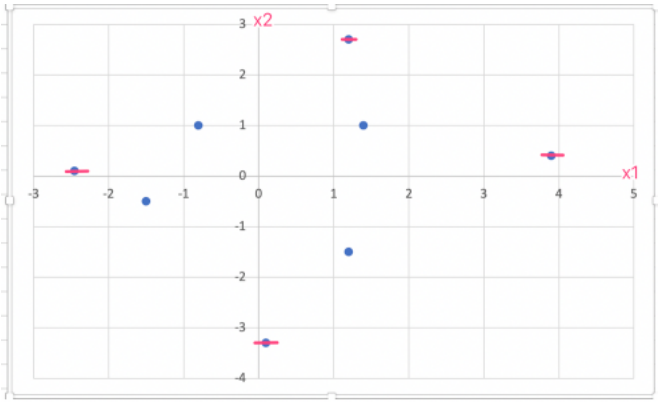

(a) Because $\hat{y}_i = w_0 + w_1 x_i$ $\hat{y}_i = w^T x_i$
therefore: $w^T \cdot x = w_0 + w_1 x_i$
$L_c(y_i, \hat{y}_i) = [\sqrt{\frac{1}{c^2}(y_i - w_0 - w_1 x_i)^2 + 1} - 1]$
$\frac{\partial L_c(y_i, \hat{y}_i)}{\partial w_0} = \frac{\partial \sqrt{\frac{1}{c^2}(y_i - w_0 - w_1 x_i)^2 + 1}}{\partial w_0}$
$= \frac{1}{2}[\frac{1}{c^2}(y_i - w_0 - w_1 x_i)^2 + 1]^{-\frac{1}{2}} \times \frac{1}{c^2} \times 2 \cdot (y_i - w_0 - w_1 x_i)^{1} \times (-1)$
$= \frac{-(y_i - w_0 - w_1 x_i)}{c^2 [\frac{1}{c^2}(y_i - w_0 - w_1 x_i)^2 + 1]^{\frac{1}{2}}}$
$\frac{\partial L_c(y_i, \hat{y}_i)}{\partial w_1} = \frac{\partial \sqrt{\frac{1}{c^2}(y_i - w_0 - w_1 x_i)^2 + 1}}{\partial w_1}$
$= \frac{1}{2}[\frac{1}{c^2}(y_i - w_0 - w_1 x_i)^2 + 1]^{-\frac{1}{2}} \times \frac{2}{c^2} \cdot (y_i - w_0 - w_1 x_i)^{1} \cdot (-x_i)$
$= \frac{(y_i - w_0 - w_1 x_i) \cdot (-x_i)}{c^2 [\frac{1}{c^2}(y_i - w_0 - w_1 x_i)^2 + 1]^{\frac{1}{2}}}$

(b)


(b) while $\frac{\partial L(y_i, \hat{y}_i)}{\partial w_0} \neq 0$ and $\frac{\partial L(y_i, \hat{y}_i)}{\partial w_0} \neq 0$
$w_0 = w_0^{(t+1)}$
$w_1 = w_1^{(t+1)}$

Q3:

(1)



The data was plot above, obviously, the +1 and -1 cannot be classified by linear regression.
Suppose we set m=0, d=2,
Therefore $k(x,y) = (x^T y)^2$
I am not sure m and d right or not, just to calculcate the question below

(2)

By selecting the vector (w, x), the transformed date is linearly separable.

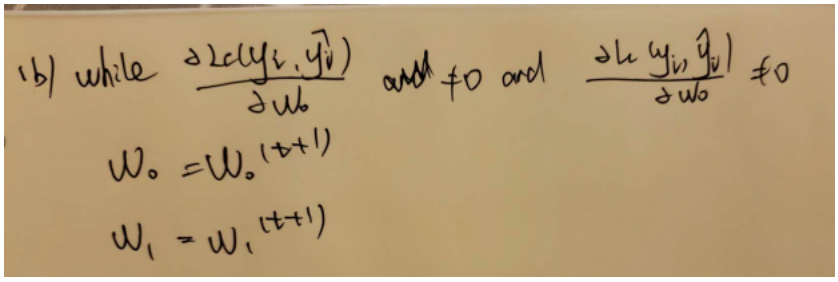
(3)

The code shows below, this still have some bugs, the code can not converge when restart iterations. The main idea of my model is to multi vector to make data linearly separate and compute wTx and ywTx, use counter to check if all the cases have been converged, if counter ==8 which means all data converged.

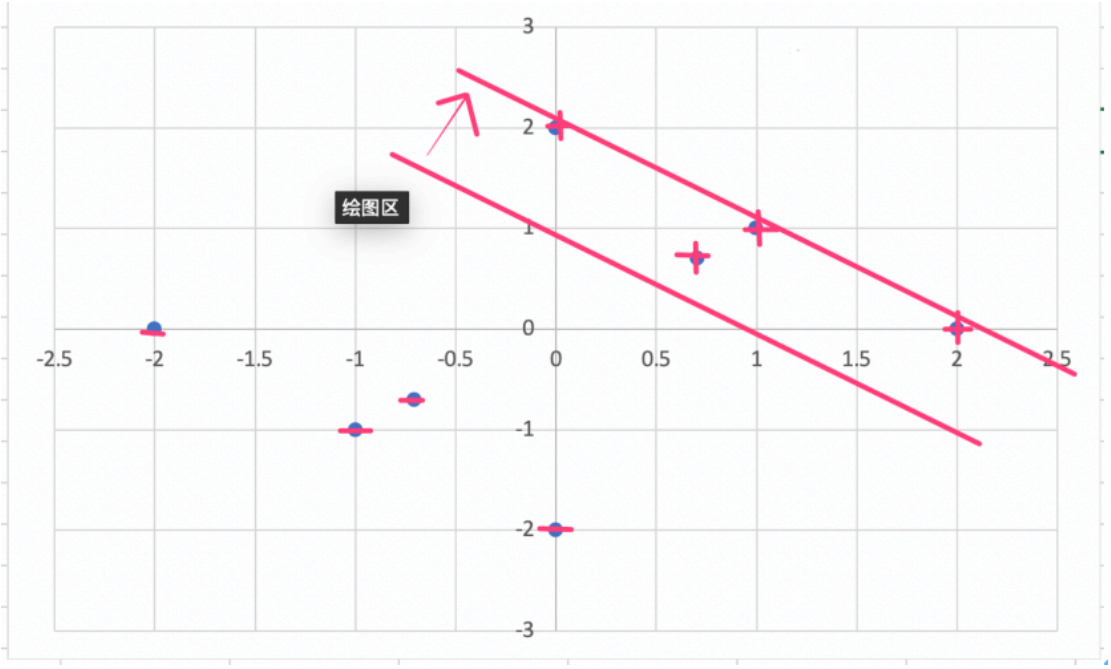
```
np_x0 = np.asarray(x0)
np_x1 = np.asarray(x1)
np_x2 = np.asarray(x2)
np_y = np.asarray(y)
w_start = [1.00,1.00,1.00]
learning_rate = 0.2
counter = 0

def check_positive(x0, x1, x2, y):
    print("--------------------")
    global w_start, counter
    wTx = (w_start[0] * x0) + (w_start[1] * x1) + (w_start[2] * x2)
    ywTx = wTx * y

    if ywTx >= 0:
        counter += 1
    else:
        w_start = [w_start[0] - y * (learning_rate * x0), w_start[1] - y * (learning_rate * x1), w_start[2] - y * (learning_rate * x2) ]

print("φ(xi)        yiφT(xi)w*")
counter2 = 0
while True:
    counter = 0
    for i in range(8):
        check_positive(np_x0[i],np_x1[i], np_x2[i], np_y[i])
        list_w = [w_start[0], w_start[1], w_start[2]]

        if list_w:
            b = "ri > 0"
        print(list_w, b )
    if counter == 8:
        break
```

```
φ(xi)        yiφT(xi)w*
--------------------
[1.0, 1.0, 1.0] ri > 0
--------------------
[1.2, 1.78, 1.08] ri > 0
--------------------
[1.2, 1.78, 1.08] ri > 0
--------------------
[1.2, 1.78, 1.08] ri > 0
--------------------
[1.4, 2.02, 1.62] ri > 0
--------------------
[1.4, 2.02, 1.62] ri > 0
--------------------
[1.2, 2.3200000000000003, 1.7200000000000002] ri > 0
--------------------
[1.2, 2.3200000000000003, 1.7200000000000002] ri > 0
```

(d)



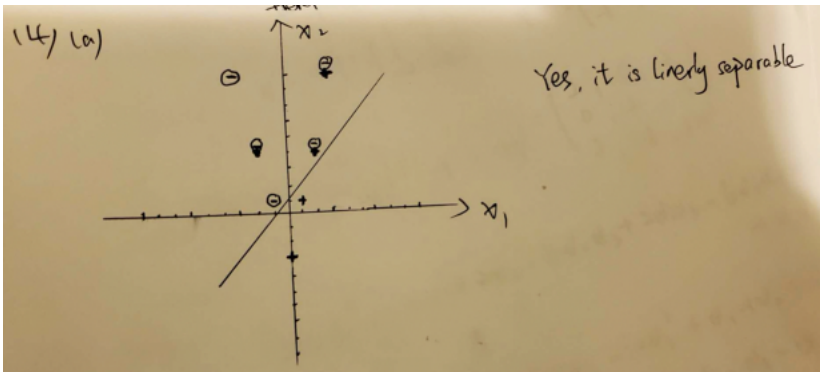
(e)



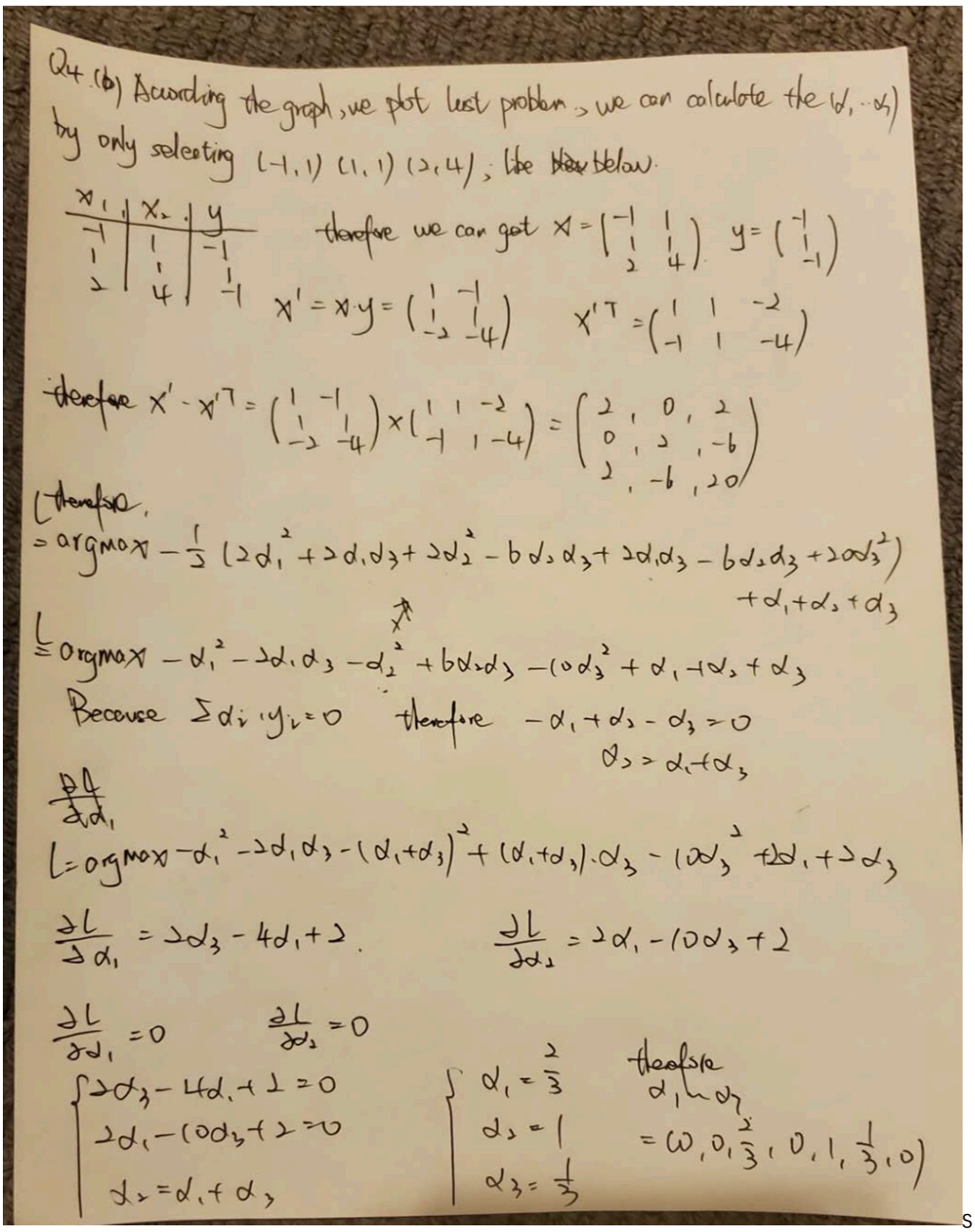
I agree with the claim, by plotting the data, it is obviously that when we move the line by applying the learning rate = 1, the most mistakes it can made is show as figure is 4. Which means make 5 mistakes during training will never happen.

Q4:

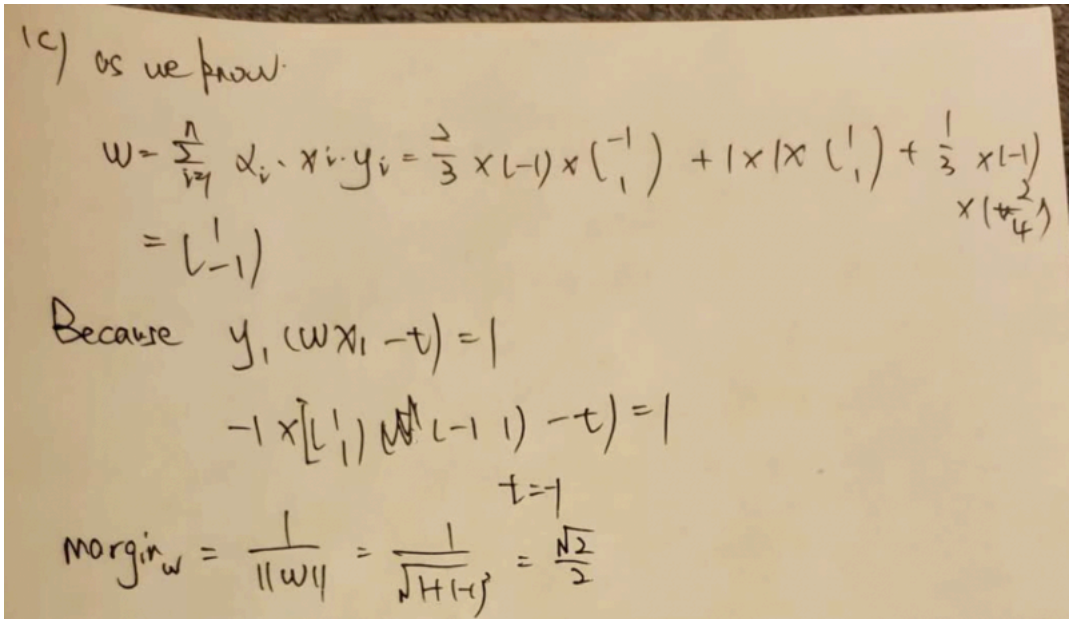
(a)


(4) (a)
x_2
x_1
Yes, it is linerly separable

(b)


Q4 (b) According the graph, we plot best problem, we can calculate the $(\alpha_1 \cdots \alpha_3)$ by only selecting $(-1,1)$ $(1,1)$ $(2,4)$, like ~~blue~~ below.
x_1 x_2 y
-1 1 -1
1 1 -1
2 4 1
therefore we can get $X = \begin{pmatrix} -1 & 1 \\ 1 & 1 \\ 2 & 4 \end{pmatrix}$ $y = \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}$
$X' = X \cdot y = \begin{pmatrix} 1 & -1 \\ 1 & 1 \\ -2 & -4 \end{pmatrix}$ $X'^T = \begin{pmatrix} 1 & 1 & -2 \\ -1 & 1 & -4 \end{pmatrix}$
therefore $X' \cdot X'^T = \begin{pmatrix} 1 & -1 \\ 1 & 1 \\ -2 & -4 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & -2 \\ -1 & 1 & -4 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 2 \\ 0 & 2 & -6 \\ 2 & -6 & 20 \end{pmatrix}$
therefore,
$= \text{argmax} -\frac{1}{2}(2\alpha_1^2 + 2\alpha_1\alpha_3 + 2\alpha_2^2 - 6\alpha_2\alpha_3 + 2\alpha_1\alpha_3 - 6\alpha_2\alpha_3 + 20\alpha_3^2) + \alpha_1 + \alpha_2 + \alpha_3$
$= \text{argmax} -\alpha_1^2 - 2\alpha_1\alpha_3 - \alpha_2^2 + 6\alpha_2\alpha_3 - 10\alpha_3^2 + \alpha_1 + \alpha_2 + \alpha_3$
Because $\sum \alpha_i \cdot y_i = 0$ therefore $-\alpha_1 + \alpha_2 - \alpha_3 = 0$
$\alpha_2 = \alpha_1 + \alpha_3$
$\frac{\partial L}{\partial \alpha_1}$
$L = \text{argmax} -\alpha_1^2 - 2\alpha_1\alpha_3 - (\alpha_1 + \alpha_3)^2 + (\alpha_1 + \alpha_3)\cdot\alpha_3 - 10\alpha_3^2 + 2\alpha_1 + 2\alpha_3$
$\frac{\partial L}{\partial \alpha_1} = 2\alpha_3 - 4\alpha_1 + 2$ $\frac{\partial L}{\partial \alpha_2} = 2\alpha_1 - 10\alpha_3 + 2$
$\frac{\partial L}{\partial \alpha_1} = 0$ $\frac{\partial L}{\partial \alpha_2} = 0$
$\begin{cases} 2\alpha_3 - 4\alpha_1 + 2 = 0 \\ 2\alpha_1 - 10\alpha_3 + 2 = 0 \\ \alpha_2 = \alpha_1 + \alpha_3 \end{cases}$ $\begin{cases} \alpha_1 = \frac{2}{3} \\ \alpha_2 = 1 \\ \alpha_3 = \frac{1}{3} \end{cases}$
therefore $\alpha_1 \sim \alpha_7 = (0, 0, \frac{2}{3}, 0, 1, \frac{1}{3}, 0)$

(c)



(d)

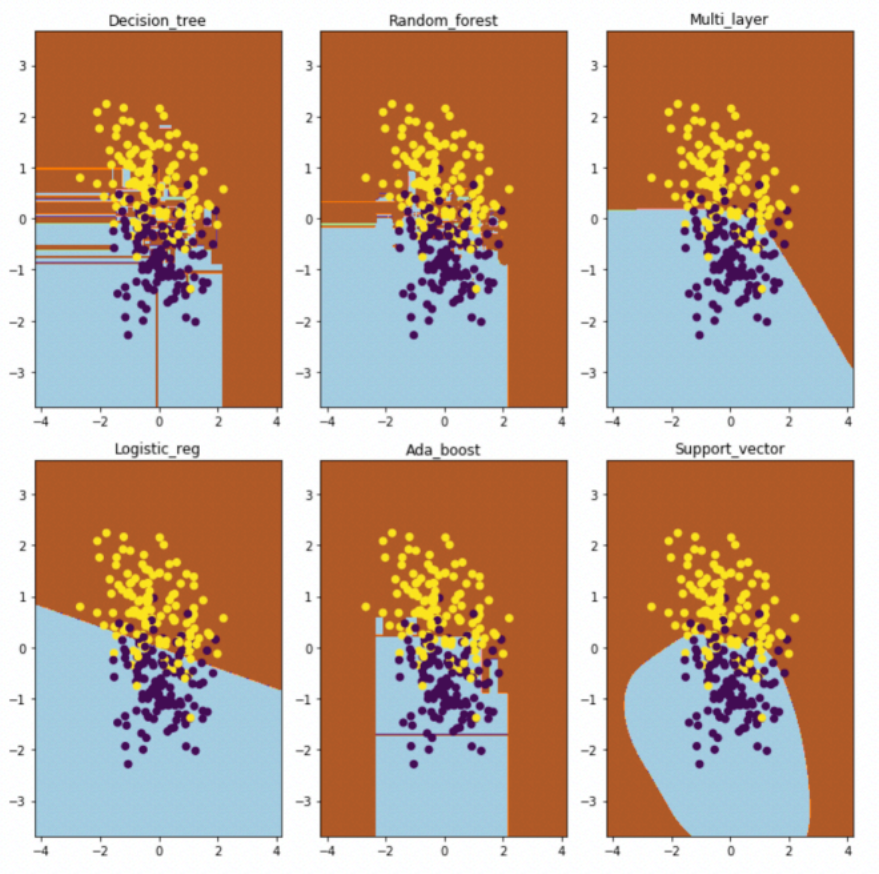
As we know, linear classifier uses a linear combination of features to make classification decisions, it can only distinguish between positive and negative samples. Most of non-linear function ,the positive and negative samples mix together, therefore Linear classifiers are unable to represent non-linear functions.

(e)

Because linearity is inseparable in low-dimensional space, we can improve the probability of linearly separable by mapping it to high-dimensional space. Especially, it can solve the possibility of each result very close to each other, Therefore, we need to map indivisible data from low-dimensional space to high-latitude space to increase accuracy.

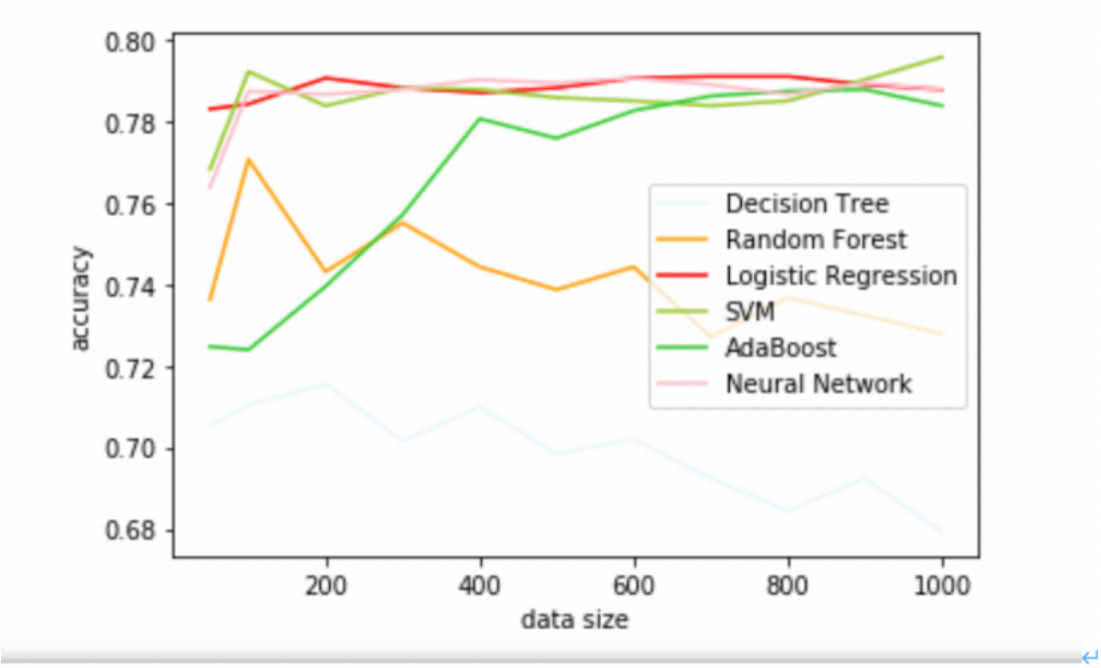
Q5:

(a)



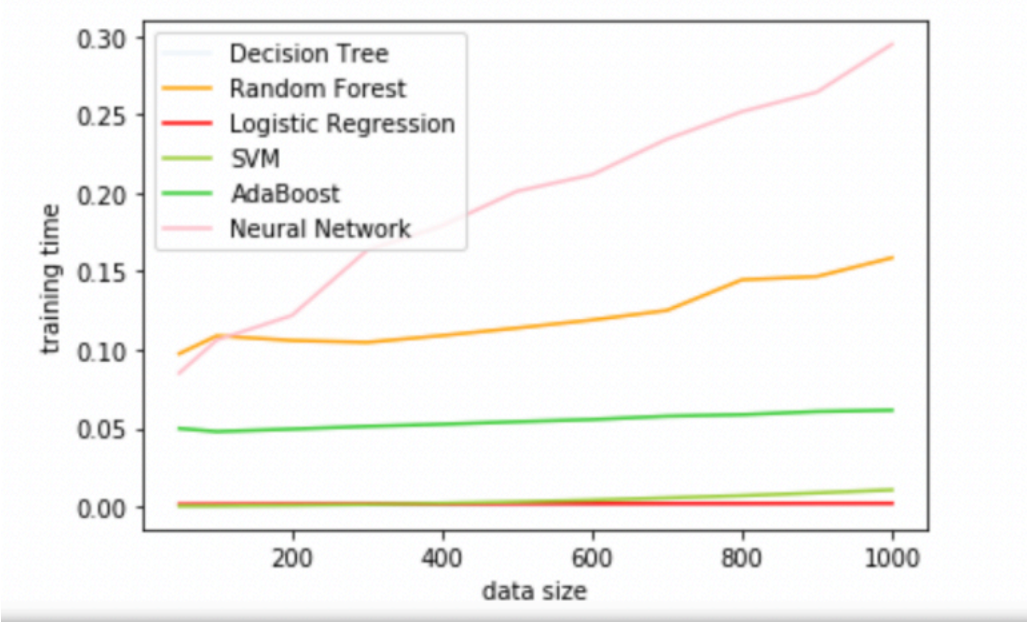
The dataset and decision boundaries show above.

(b)



The accuracy of each model show above, Obviously, most of model except decision tree ‘s accuracy is increasing with the larger data size. After 400, random forest and Decision tree keep decreasing. At the size of 600+, logistic regression, SVM, Neural network, and Adaboost keep stability at rate of 0.75, much better than Decision tree and random forest. (the color shown above may not right, because I am a little bit color blindness, sorry for that==)

(c)



According the figure shown above, Random tree and neural network increasing significantly, in contract, Adaboost, logistic regression and SVM keep at a lower time. According to the accuracy mentioned in the last problem, the SVM and Adaboost is much better to use binary classification at some certain situation.