Answer for writing part

Q1(a):

Q1:

(a):

	X_iA	X_iB	X_iC	X_iD		
d_1	2	0	4	4		
d_2	0	3	3	0		
d_3	3	0	0	2	+	$P(+)=0.5$
d_4	0	0	2	0		
d_5	0	0	0	1		
d_6	3	0	0	0		
d_7	4	3	0	0	−	$P(-)=0.5$
d_8	4	0	0	1		

Fora Multinomial Naive Bayes:

$X|C_+ \sim \text{Multinom}(n, K, \theta_A, \theta_B, \theta_C, \theta_D)$ θ is probability

$n = 23$, $K=4$

$\theta_A^+ = \frac{5}{23}$ $\theta_B^+ = \frac{3}{23}$ $\theta_C^+ = \frac{9}{23}$ $\theta_D^+ = \frac{6}{23}$

$X|C_- \sim \text{Multinomial}(n, k, \theta_A, \theta_B, \theta_C, \theta_D)$:

$n = 16$ $K = 4$

$\theta_A^- = \frac{11}{16}$ $\theta_B^- = \frac{3}{16}$ $\theta_C^- = 0$ $\theta_D^- = \frac{2}{16}$

Accordy to the formula: $x_*(1,0,0,2)$

$$c_* = \underset{k(+,-)}{\arg\max}\ p(C_k)\,p(x_* | C_k)$$

$$P(x_*|+)\,\cancel{P(\ )} = \frac{3!}{1! \times 0! \times 0! \times 2!} \cdot \left(\frac{5}{23} \times \left(\frac{3}{23}\right)^0 \times \left(\frac{9}{23}\right)^0 \times \left(\frac{6}{23}\right)^2\right)$$

$$= 3 \times \left(\frac{180}{12167}\right)$$

$$= 0.04438$$

However For $P(x_*|-)$:

$$\frac{3!}{1!0!0!2!}\left(\frac{11}{16}\times\left(\frac{3}{16}\right)^0\times(0)\times\left(\frac{2}{16}\right)^2\right)$$

$$=0$$

this is wrong, the value will always be 0

therefore argMax always being +.

And. For Baye's Rule:

$$P(C_k|x)=\frac{P(x|C_k)P(C_k)}{P(x)}$$

$$P(x)=\sum_{k=1}^{K}P(x|C_k)P(C_k)$$

so:

$$P(+|x_*)=\frac{P(x_*|+)\cdot P(+)}{P(x_*|+)\cdot P(+)+P(x_*|-)\cdot P(-)}$$

$$=\frac{P(x_*|+)\cdot P(+)}{P(x_*|+)\cdot P(+)}$$

$$=1$$

Without Smoothing, the model will always predict +, this is not correct.

Q1(b):

(b): Multinomial Smoothing:

New θ_i^+: $\quad x_*(1,0,0,2)$

$\theta_A^+ = \frac{5+1}{23+4} = \frac{6}{27} \quad \theta_B^+ = \frac{4}{27}$

$\theta_C^+ = \frac{10}{27} \quad \theta_D^+ = \frac{7}{27}$

New θ_j^-:

$\theta_A^- = \frac{12}{20} \quad \theta_B^- = \frac{4}{20}$

$\theta_C^- = \frac{1}{20} \quad \theta_D^- = \frac{3}{20}$

$$P(x_*|+) = \frac{3!}{1!0!0!2!} \cdot \left(\frac{6}{27} \times \left(\frac{4}{27}\right)^0 \times \left(\frac{10}{27}\right)^0 \times \left(\frac{7}{27}\right)^2\right)$$

$$= 0.04481$$

$$P(x_*|-) = \frac{3!}{1!0!0!2!} \cdot \left(\frac{12}{20} \times \left(\frac{4}{20}\right)^0 \times \left(\frac{1}{20}\right)^0 \times \left(\frac{3}{20}\right)^2\right)$$

$$= 0.0405$$

Follow steps in (a):

$$P(-|x_*) = \frac{P(x_*|-) \cdot P(-)}{P(x_*|-) \times P(-) + P(x_*|+) \times P(+)}$$

$$= 0.4747$$

which means give x_*, the propobabity it is + is 0.4747

Q1(c):

(C)

	A	B	C	D	
1	1	0	1	1	+
2	0	1	1	0	+
3	1	0	0	1	+
4	0	0	1	0	+
5	0	0	0	1	−
6	1	0	0	0	−
7	1	1	0	0	−
8	1	0	0	1	−

piror: $P(c+) = P(c-) = \frac{1}{2}$.

$P_A^+ = \frac{2}{4}$ $P_B^+ = \frac{1}{4}$ $P_C^+ = \frac{3}{4}$ $P_D^+ = \frac{2}{4}$

$P_A^- = \frac{3}{4}$ $P_B^- = \frac{1}{4}$ $P_C^- = 0$ $P_D^- = \frac{2}{4}$

classifier $C_* = \underset{K\in(+,-)}{\arg\max}\, P(C_K)\, P(x_* | C_K)$

$x_* = (1, 0, 0, 2) = (1, 0, 0, 1)$

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

$$= \frac{1}{2} \times \frac{2}{4} \times \frac{3}{4} \times \frac{1}{4} \times \frac{2}{4}$$

$$= \frac{3}{128}$$

$$P(C-) P(x_* | C-) = P(C-) \cdot \frac{3}{4} \times (1 - \frac{1}{4}) \times (1 - 0) \times \frac{2}{4}$$

$$= \frac{9}{64}$$

this model is not correct.

we have $P_C^- = 0$ if a x_* has $x_*C = 1$

then, regardless of the values of the other features, get $P(x_*|C-) = 0$

therefore, the model will predict + for every x_* with $x_*C = 1$.

$$P(+|x_*) = \frac{\frac{3}{128}}{\frac{3}{128} + \frac{9}{64}} = \frac{1}{7}$$

compare (a)(b), this is not correct.

f $x_*C = 0$, $P(x_c = 0|C-)$ will always being $(1-0) = 1$, the model will ↑ prob for C−, the model with out smooth can't dual with this data set.

Q1(d):

(d): new $P_{k_i}^+$:

$P_A^+ = \frac{2+1}{4+1} = \frac{3}{5}$ $\quad P_B^+ = \frac{2}{5}$

$P_C^+ = \frac{4}{5}$ $\quad P_D^+ = \frac{3}{5}$

New $P_{k_i}^-$:

$P_A^- = \cancel{\frac{3}{4}}\ \frac{4}{5}$ $\quad P_B^- = \frac{2}{5}$

$P_C^- = \frac{1}{5}$ $\quad P_D^- = \frac{3}{5}$

$$P(C+)P(x_*|C_+) = \frac{1}{2} \times \frac{3}{5} \times \left(1-\frac{2}{5}\right) \times \left(1-\frac{4}{5}\right) \times \frac{3}{5} = \frac{27}{1250}$$

$$P(C-)P(x_*|C_+) = \frac{1}{2} \times \frac{4}{5} \times \left(1-\frac{2}{5}\right) \times \left(1-\frac{1}{5}\right) \times \frac{3}{5} = \frac{72}{625}$$

$$P(-|x_*) = \frac{\frac{72}{625}}{\frac{72}{625} + \frac{27}{1250}} = 0.8421$$

the $P(-|x_*) = 0.8421$

Q2:

(a)

$$\begin{aligned}
\frac{\partial \mathcal{L}(y_i,\hat{y_i})}{\partial w_0} &= \frac{\partial \mathcal{L}(y_i,\hat{y_i})}{\partial(\frac{1}{c^2}(y_i - w^T X_i)^2 + 1)} \cdot \frac{\partial(\frac{1}{c^2}(y_i - w^T X_i)^2 + 1)}{\partial(y_i - w^T X_i)} \cdot \frac{\partial(y_i - w^T X_i)}{\partial w_0} \\
&= \frac{1}{2\sqrt{\frac{1}{c^2}(y_i - w^T X_i)^2 + 1}} \cdot \frac{2}{c^2}(y_i - w^T X_i) \cdot (-1) \\
&= \frac{-(y_i - w^T X_i)}{\sqrt{c^2(y_i - w^T X_i)^2 + c^4}}
\end{aligned} \tag{9}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}(y_i,\hat{y_i})}{\partial w_1} &= \frac{\partial \mathcal{L}(y_i,\hat{y_i})}{\partial(\frac{1}{c^2}(y_i - w^T X_i)^2 + 1)} \cdot \frac{\partial(\frac{1}{c^2}(y_i - w^T X_i)^2 + 1)}{\partial(y_i - w^T X_i)} \cdot \frac{\partial(y_i - w^T X_i)}{\partial w_0} \\
&= \frac{1}{2\sqrt{\frac{1}{c^2}(y_i - w^T X_i)^2 + 1}} \cdot \frac{2}{c^2}(y_i - w^T X_i) \cdot (-x_i) \\
&= \frac{-x_i(y_i - w^T X_i)}{\sqrt{c^2(y_i - w^T X_i)^2 + c^4}}
\end{aligned} \tag{10}$$

b:)

(b): Initialize each w_i to small random value.

while (not convergent):

$\nabla w_j = 0$

For x_i in training set:

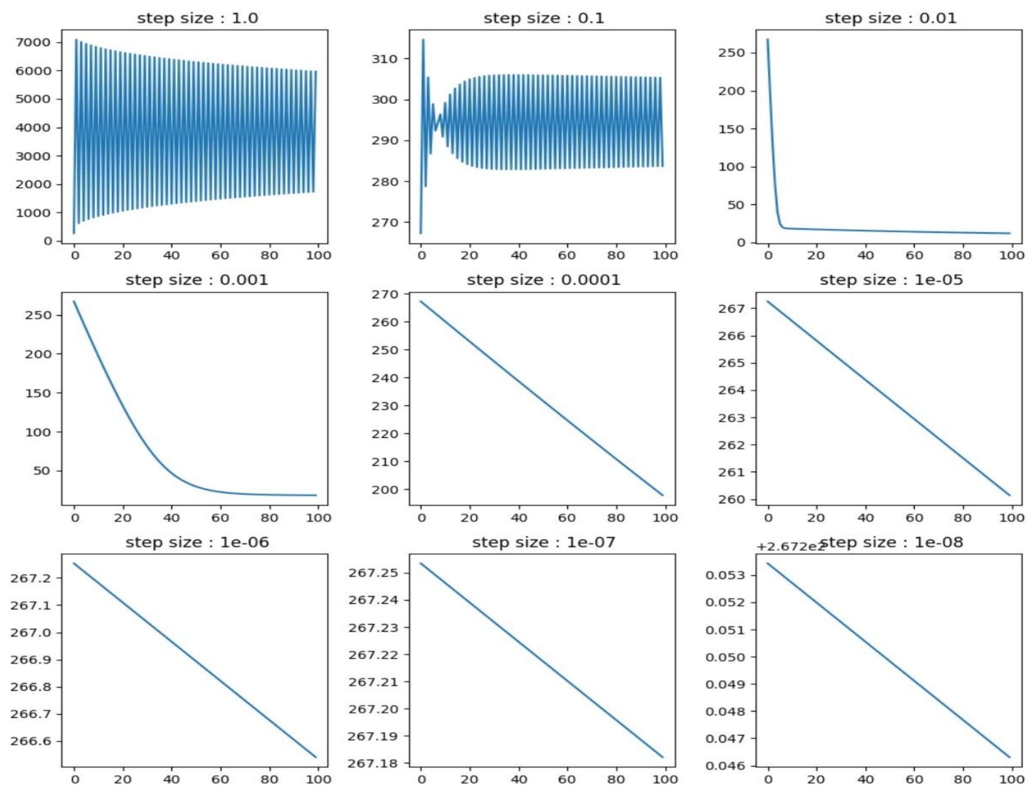
$y_i = f(x_i)$

for each w_i:

$$\nabla w_j = \nabla w_j - \alpha \frac{\partial L}{\partial w_j}$$

$w_j = w_j + \nabla w_j$.

C) :

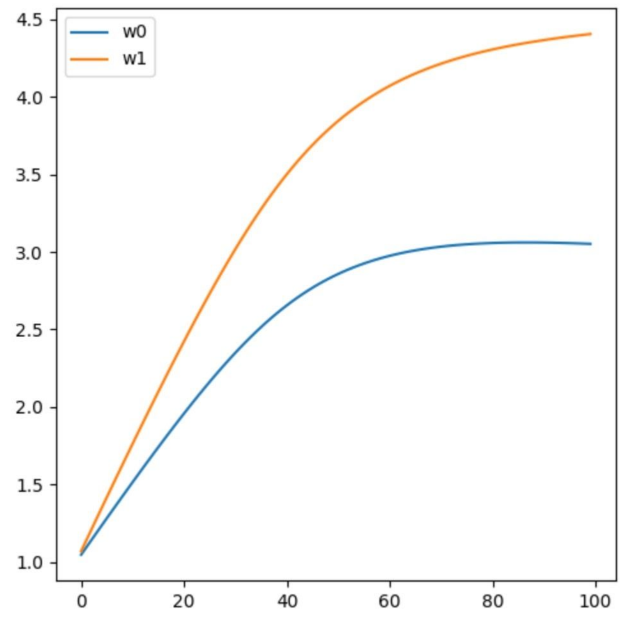


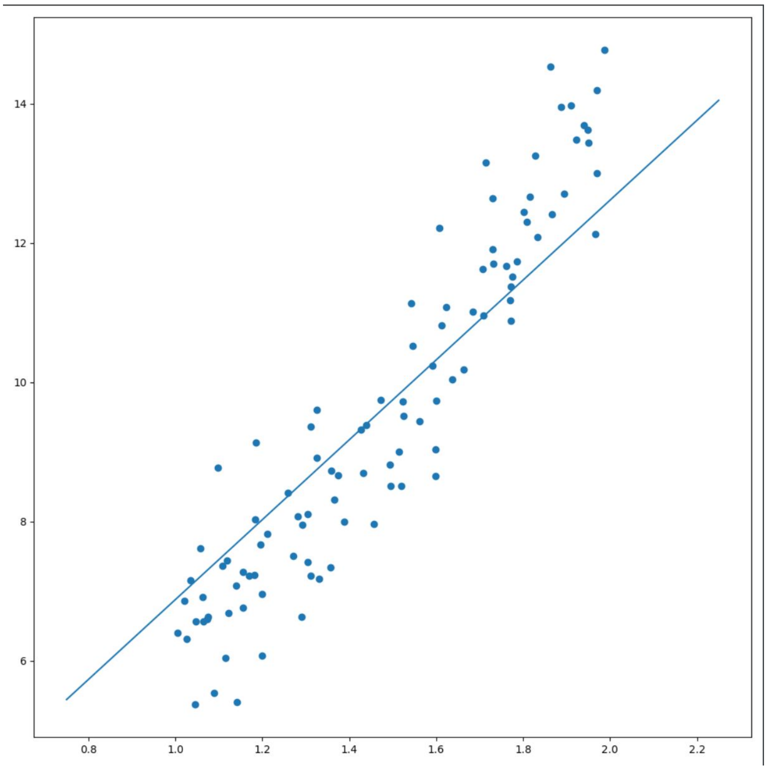

```python
def compute_loss(x, y, w, c):
    loss = np.sum(np.sqrt((1.0/c**2)*(y-np.matmul(w, x))**2+1)-1)
    return loss


def compute_gradient(x, y, w, c):
    grad = np.sum(x * (-(y-np.matmul(w, x)) / np.sqrt(c**2*(y-y_pred)**2+c**4)), axis=1)
    return grad


def gradient_descent(x, y, alpha, iter_num=100):
    loss = []
    x_0 = np.ones(x.shape)
    x = np.concatenate((x_0[np.newaxis, :], x[np.newaxis, :]), axis=0)
    W = np.array([1, 1], dtype=np.float64)
    c = 2

    for i in range(iter_num):
        loss.append(compute_loss(x, y, W, c))
        W = W - alpha * compute_gradient(x, y, W, c)

    return loss


if __name__ == '__main__':
    alphas = [10e-1, 10e-2, 10e-3, 10e-4, 10e-5, 10e-6, 10e-7, 10e-8, 10e-9]
    losses = []
    for i in range(len(alphas)):
        losses.append(gradient_descent(x, y, alphas[i]))
        print(losses[i])
```

d:)

when step size > 0.01, loss oscillates within a certain range during the iteration; when step size less than 0.001, loss oscillates in a very small range. step size have a relationship with the update magnitude of the parameter, too big or too small, the model will not converge e:)

there are some models with outlier Deviating from the fitting. We can try to increase the parameters of the model to enhance the fitting ability. y = w0+ w1x+w1x^2

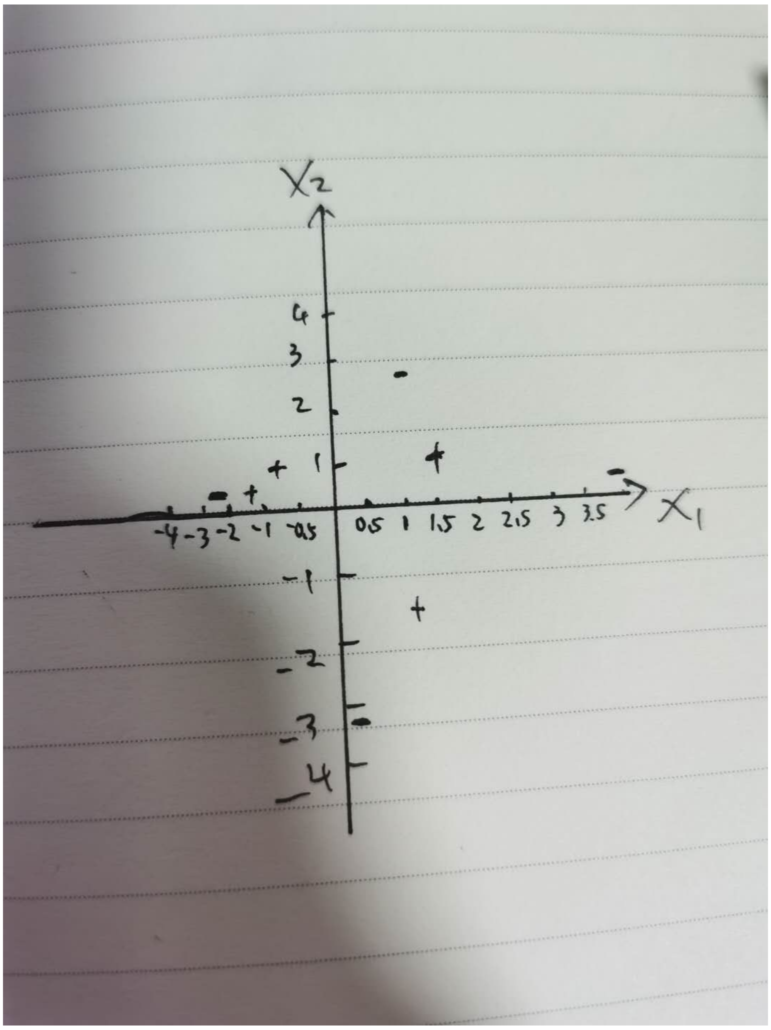




f): In the case that the model can converge, the larger the step size is, the faster the convergence will be, and we can choose to make optimal step size to the maximum c value.

Q3: don't have enough time for q3

a):



d):

~~3.3: (a):~~ don't have ~~time~~ enough time

d: $x=\begin{pmatrix}x_1\\x_2\end{pmatrix}$ $y=\begin{pmatrix}y_1\\y_2\end{pmatrix}$

$$K(x,y)=2(x^Ty+3)^3$$

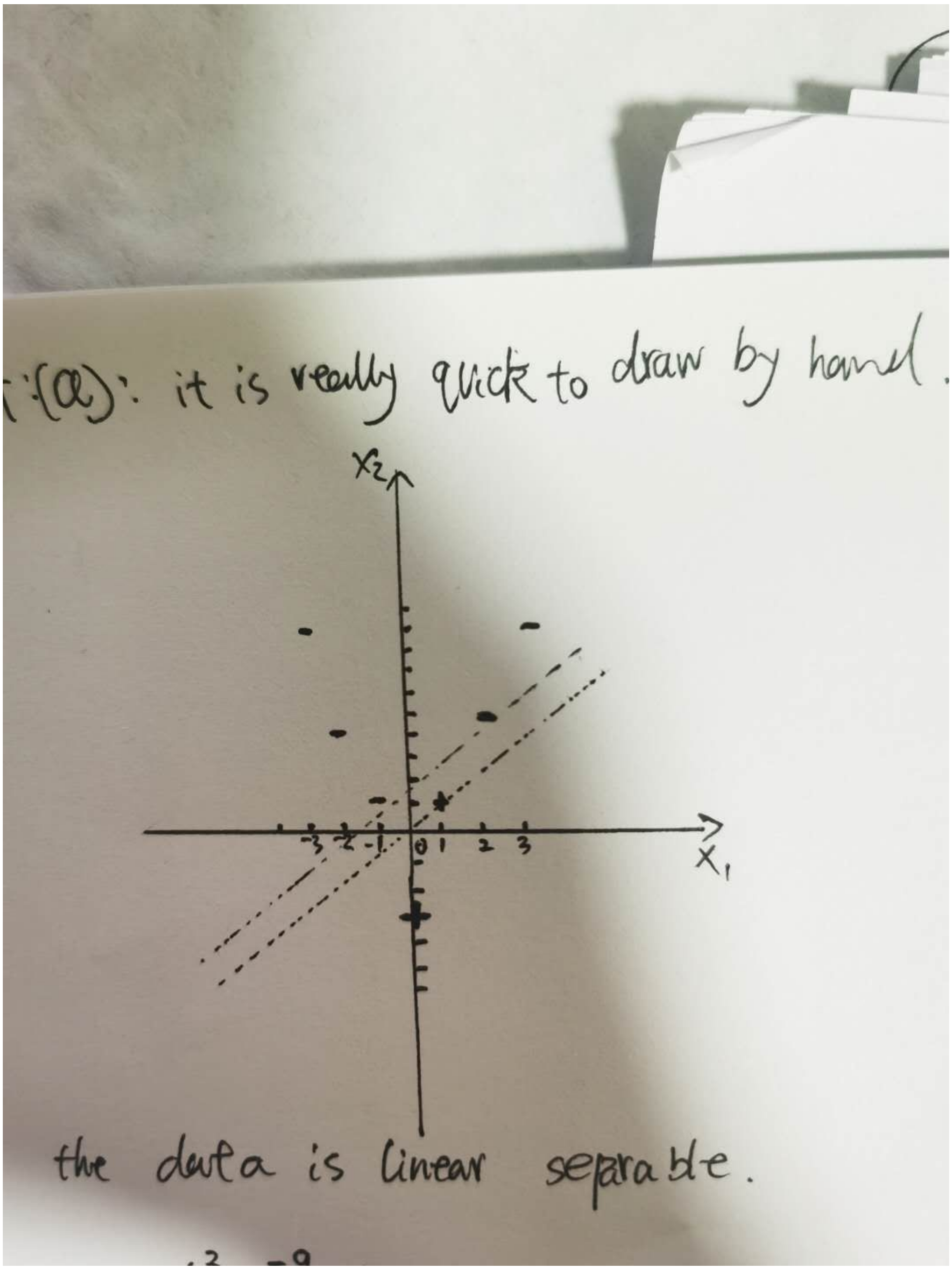
it is easy to get

$$\phi(x)=\begin{pmatrix}x^3\\x^2\\x\\const\end{pmatrix} \quad y=\begin{pmatrix}y^3\\y^2\\y\\const\end{pmatrix}$$

e:

Q4:

a):


(a): it is really quick to draw by hand.
X_2
X_1
-3 -2 -1 0 1 2 3
the data is linear separable.

b):

(b): from graph (a). I choose support vector

$x_1(-1,1)$ $y=-1$

$x_2(1,1)$ $y=+1$ $\quad X=\begin{pmatrix}-1&1\\2&4\\1&1\end{pmatrix}$ $y=\begin{pmatrix}-1\\-1\\+1\end{pmatrix}$

$x_3(2,4)$ $y=-1$

$x'\begin{pmatrix}1&-1\\-2&-4\\1&1\end{pmatrix}$ $\quad (x')^T=\begin{pmatrix}1&-2&1\\-1&-4&1\end{pmatrix}$

so: $\cancel{\underset{d_1 d_2 d_3}{\arg\mathrm{Max}} -\frac{1}{2}(}$ $\quad (x')(x')^T=\begin{pmatrix}2&2&0\\2&20&-6\\0&-6&2\end{pmatrix}$

$$\arg\mathrm{Max} -\frac{1}{2}(2d_1^2+2d_1d_2+0+2d_2d_1+20d_2^2-6d_2d_3+0-6d_3d_2+2d_3^2)+d_1+d_2+d_3$$

$$=\arg\mathrm{Max} -\frac{1}{2}(2d_1^2+4d_1d_2-12d_2d_3+20d_2^2+2d_3^2)+d_1+d_2+d_3$$

s.t. $d_1\geq 0$ $d_2\geq 0$, $d_3\geq 0$, $d_1+d_2=d_3$

let $d_3=d_1+d_2$:

$$\arg\mathrm{Max} -\frac{1}{2}(2d_1^2+4d_1d_2-12d_1d_2-12d_2^2+20d_2^2+2(d_1+d_2)^2)+\cdots,$$

$$=\arg\mathrm{Max} -\frac{1}{2}(4d_1^2-4d_1d_2+10d_2^2)+2d_1+2d_2=L(d_1,d_2)$$

$$\frac{\partial L}{\partial d_1}=-4d_1+2d_2+2=0$$

$$\frac{\partial L}{\partial d_2}=2d_1-10d_2+2=0$$

$$\begin{cases}-4d_1+2d_2+2=0\\2d_1-10d_2+2=0\end{cases}$$

$d_1=\frac{2}{3}$ $\quad d_2=\frac{1}{3}$ $\quad d_3=1.$

c):

(c):

$$x = \begin{pmatrix} -1 & 1 \\ 2 & 4 \\ 1 & 1 \end{pmatrix} \quad y = \begin{pmatrix} -1 \\ -1 \\ +1 \end{pmatrix}$$

$$W = \sum_{i=1}^{n} a_i y_i x_i$$

$$w = \cancel{0} - \frac{2}{3}x_1 - \frac{1}{3}x_2 + x_3$$

$$W = \begin{pmatrix} \frac{2}{3} + (-\frac{2}{3}) + (1) \\ -\frac{2}{3} + (-\frac{4}{3}) + (1) \end{pmatrix}$$

$$W = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

t can be obtaind from any support vector

set X_3, since $y_3(w \cdot x_3 - t) = 1$

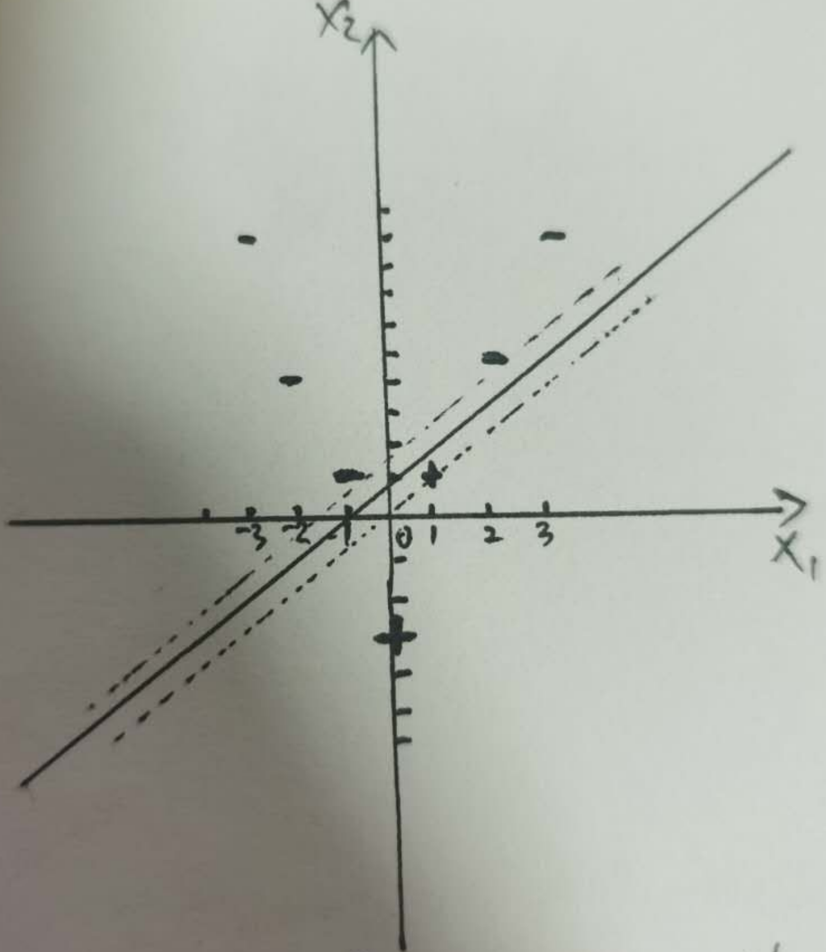
$$1 \cdot \left(\begin{pmatrix} 1 \\ -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} - t \right) = 1$$

$$(0 - t) = 1$$

$$t = -1$$

$$\text{margin} = \frac{1}{\|W\|} = \frac{1}{\sqrt{1^2 + (-1)^2}} = \frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2}$$

:(a): it is really quick to draw by hand



the data is linear separable.

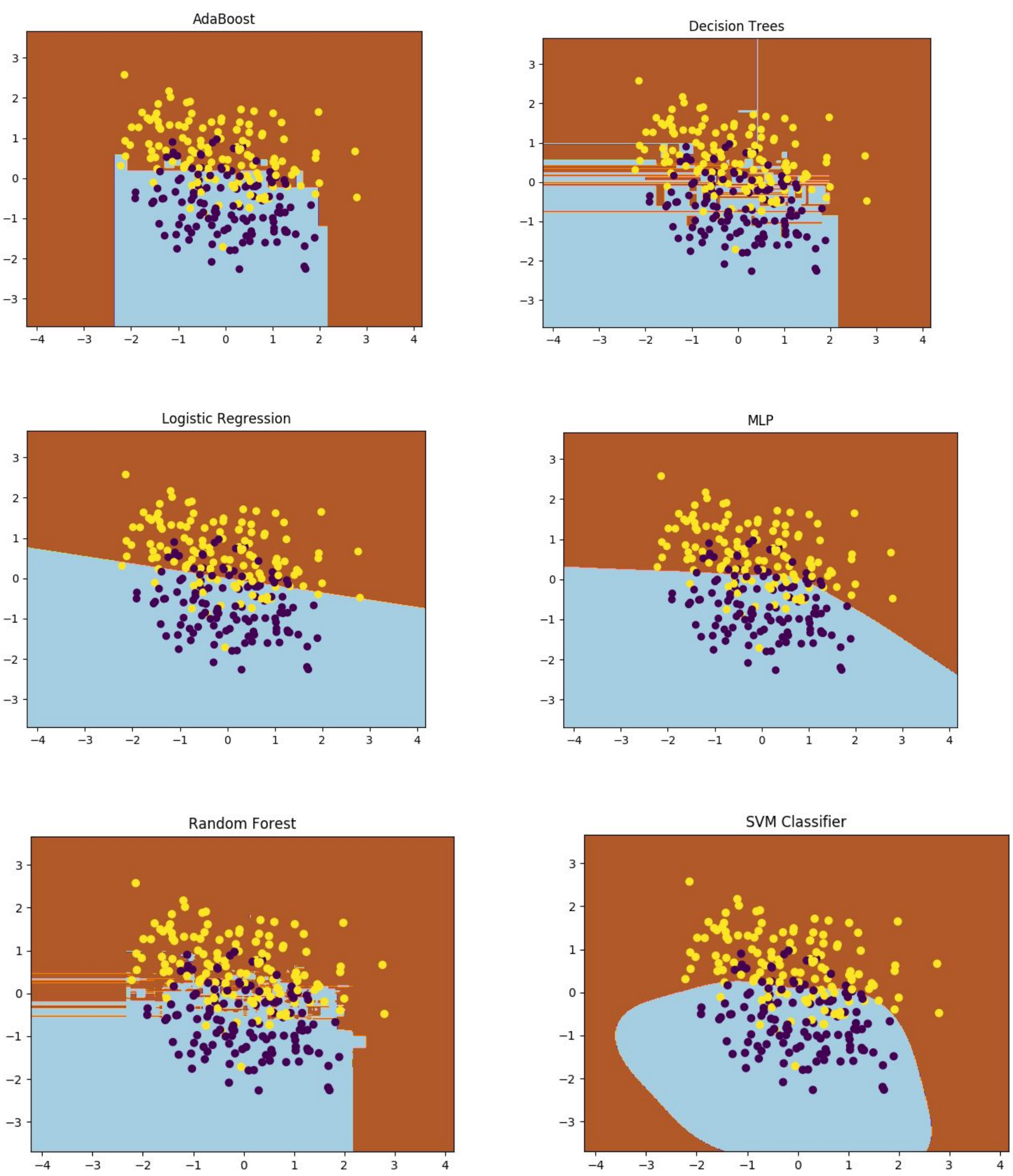
$x' = \begin{pmatrix} 3 & -9 \\ 2 & -4 \end{pmatrix}$ $(x')^T = \begin{pmatrix} 3 & 2 & 1 & 0 \end{pmatrix}$

d): In short, a linear classifier can be seen as a multidimensional line (hyperplane), like a knife that divides a space into two. If the equation is a curve or if space is distorted. So this line doesn't perfectly divide the data into two parts. Just like the example, perceptron can’t learn XOR

e:)we don’t want dot product between feature vectors so we choose Kernel trick and it allows us to calculate faster. It allows us to operate in the original feature space without computing the coordinates of the data in a higher-dimensional space from the tutorial, we see how expensive to calculate K function, so kernel trick make algorithm work more efficiently.

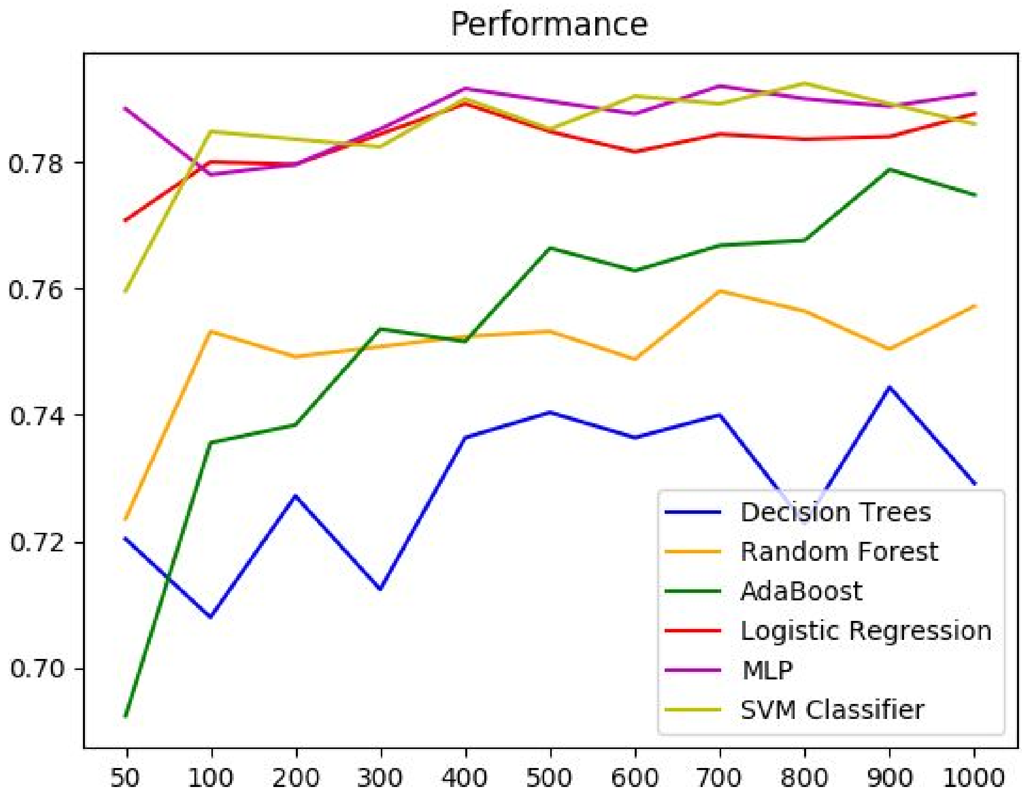
Q5:

a):


AdaBoost
Decision Trees
Logistic Regression
MLP
Random Forest
SVM Classifier

b): All classifiers but DT tend to achieve higher accuracy as the training set gets larger. Among them, AdaBoost gains the greatest improvement, while DT model's performance stays changelessly. Besides, non-tree-based models(Logistic Regression, MLP, SVC) always perform better than a tree-based models(DT,RF,AdaBoost) in this experiment

Tree-based models are less robust and are likely to overfit to the training set, so they performed badly than non-tree-based models



c): The training time of MLP significantly increases as the training set becomes larger, while the remaining five classifiers are not sensitive to the training set size.

Regarding the prolonged training time of MLP classifier, it may be the low learning rate in SGD optimizer to blame.