EE559
HW6
1

(a)
I uesd scikit-learn and Python

(b)
Normalizing factors should be calculated from the training data only because we try to train these two models (perceptron and linear regression) by training data rather than testing data. Normalizing training data is to get a better model, but testing data is to test whether is a good model or not. It should not be normalized.
I used StandardScaler from sklearn, the algebra behind it is x' = (x – mean) / standard deviation(making x' centered into mean zero and standard deviation one)
Scaler.mean_(feature's mean):

```
[1.29653933e+01 2.27000000e+00 2.37629213e+00 1.96494382e+01
 9.89101124e+01 2.27235955e+00 2.02943820e+00 3.60674157e-01
 1.57617978e+00 5.09123596e+00 9.53213483e-01 2.56033708e+00
 7.29707865e+02]
```

standard deviation:

```
[8.19560112e-01 1.10306417e+00 2.73004021e-01 3.46517583e+00
 1.15589748e+01 6.14548382e-01 9.19718276e-01 1.20241309e-01
 5.41470129e-01 2.40437795e+00 2.29907577e-01 7.23461482e-01
 3.07221225e+02]
```

(c)
(i) default initial weight vector is a zero vector([0, 0, 0, …, 0] [(D + 1)*1)])
(ii)
Halting condition:
The stopping criterion. The iterations will stop when (loss > previous_loss - tol) default tol=1e-3 and if the perceptron cannot stop based on the condition above, it will stop when running 1000 epochs.

(d)
1st two features:
Its weight (coef_ + intercept_) is :

```
[[ 2.39810751 -2.10323213 -1.        ]
 [-3.65391958 -0.82497467 -1.        ]
 [ 1.4642001  -0.39888885  0.        ]]
```

Accuracy Score of training data in 1st 2 feature: 0.797753
Accuracy Score of testing data in 1st 2 feature: 0.764045

All 13 features:
Its weight (coef_ + intercept_) is :

```
[[ 4.14486533  1.79499983  2.45952668 -3.21854057  1.1722979  -1.08730617
   3.73722161 -0.13829867  0.3894931  -3.03355811  2.81011799  2.61477002
   4.98188016 -5.        ]
 [-4.95291733 -3.07325729 -1.41908434  3.11640071 -2.69745163  1.09791047
   0.22710912  1.09237262 -0.66049895 -5.36418499  2.83777932 -0.13729251
  -3.86231215 -2.        ]
 [ 0.31354173  1.88565639  0.40827484  1.22567836  3.24860661 -0.89204083
  -4.01515234 -1.21945789 -0.46004592  4.78135303 -3.80123438 -2.15397039
  -0.07961909 -5.        ]]
Accuracy Score of training data in all features: 1.000000
Accuracy Score of testing data in all features: 0.977528
```

(e)

```
Max in 100 times 1st two features:
Its weight (coef_ + intercept_) is : [[ 2.5432341  -1.00452907 -0.31108462]
 [-2.16607237 -0.49276653 -1.39064068]
 [ 0.6623251   1.37687185 -0.33380493]]
Accuracy Score of training data 1st 2 feature MAX in 100: 0.842697
Accuracy Score of testing data 1st 2 feature MAX in 100: 0.786517
Max in 100 times all 13 features:
Its weight (coef_ + intercept_) is : [[ 2.82012475  0.62455221  0.50523242 -0.36804129
3.31799723  0.37334702
   1.74368661 -0.56983361  1.02959072 -0.17610858  1.10379236  1.27968644
   2.95691648 -2.53159728]
 [-3.69472764 -4.10302219 -3.42697268  2.33531173 -2.94605961  1.33746434
   1.27355029  3.61236291  1.42715099 -7.63252537  1.39151142 -1.40374582
  -5.93922061 -4.50977333]
 [ 2.57990354  0.36154093  0.61038789 -0.79811146  0.87582794 -2.8144044
  -2.29535919 -1.25663396 -2.26877443  5.87390538 -4.89729993 -2.8128092
   0.45662581 -6.61765895]]
Accuracy Score of training data in all features MAX in 100: 1.000000
Accuracy Score of testing data in all features MAX in 100: 0.943820
```

(f)
From (d), the accuracy in both training data and testing data are considerable improvements when we use all features comparing with 1st two features. Bec 13 features has more constraints(g(x)) when we try to get an optimal weight (minimum of J(w)), which means we get a more precise w to classify all classes. w(i) will be compared more times in one epoch.
Training data is also has a better accuracy than the testing data, because this model is trained by Training data, which means the weight is more fitter in training data set than in testing data set

From (e), like (d), the maximized accuracy in both training data and testing data are considerable improvements when we use all features comparing with 1st two features.

From (d) and(e), the (e) should have a better accuracy of training data than (d). Because it change 100 times initial weight to get a best weight to classify all classes, The error from choosing different weight vector can be decreased. Like in 1st two features, the accuracy of training data improves from 0.797753 to 0.842697. But 100 times is not too much, so sometime it may not get a perfect w0 for w than the one time. So this improvement is not stable even sometime may get a little bit smaller result. And in all features, the accuracy has already been 100% so it still be 100%(max, cannot improve)
In testing data, cause getting a better weight for the model, it should also get a better accuracy. Like in 1st two features, the accuracy of training data improves from 0.764045 to 0.786517. However, in all features, the accuracy of training data has always been 100%, so we will get a similar model, which we will get a similar accuracy of testing data(a little bit better or less)

(g)
Using unnormalized data:
Accuracy Score of testing data in 1st 2 feature unstandardized: 0.752809
Accuracy Score of testing data in all features unstandardized: 0.977528

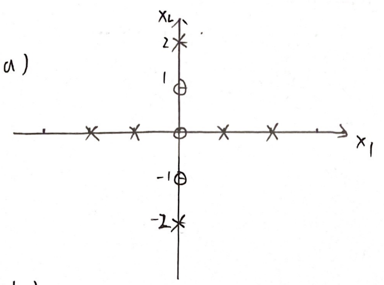
(h)
Using standardized data:
Accuracy Score of testing data in 1st 2 features standardized: 0.752809
Accuracy Score of testing data in all features standardized: 0.977528

(i)
They are identical

(j)
They are similar, and in both cases, the accuracy is a considerable improvement when we use all features rather than 1st two features.

(2)

2.
(a)



o-S_1
x-S_2.

No, they are not linearly separable.

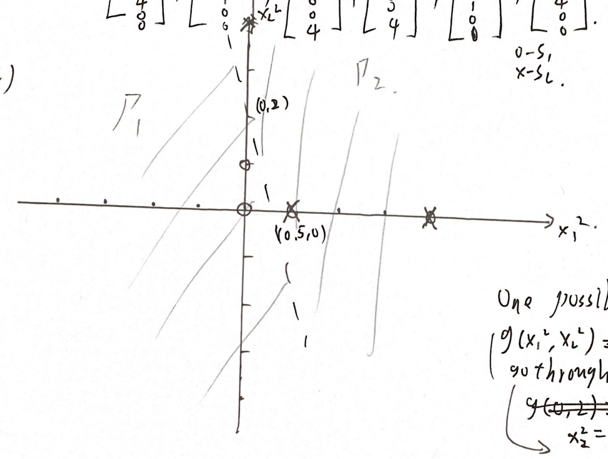
(b). $D=2$ $\underline{u} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}_{[(D'+1)\times 1]}$ $\Rightarrow D'=5$

S_1: $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

S_2 $\begin{bmatrix} 1 \\ -2 \\ 0 \\ 4 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 0 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ -2 \\ 0 \\ 0 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 0 \\ 4 \\ 0 \\ 0 \end{bmatrix}$.

o-S_1
x-S_2.

c)



One possible decision boundary ~~is~~:
($g(x_1^2, x_2^2)$ ~~$= a x_1^2 + b x_2^2$~~ $+ c = 0$)
go through (0,2), (0.5,0)
~~$g(0,2) = 2b + c$~~
$x_2^2 = k x_1^2 + b \Rightarrow \begin{cases} k=-4 \\ b=2 \end{cases}$
$g(x_1, x_2^2) = -4x_1^2 - x_2^2 + 2$.

$\Rightarrow \underline{w}' = \begin{pmatrix} 2 \\ 0 \\ 0 \\ -4 \\ 0 \\ -1 \end{pmatrix}$

Decision rule.
$g(\underline{u}) = \underline{w}^T \underline{u} \geq 0 \rightarrow \underline{u} \in S_1$
$g(\underline{u}) = \underline{w}^T \underline{u} < 0 \rightarrow \underline{u} \in S_2$.

d) back in to orlgin space

the decislon boundary is.

$g(x_1, x_2) = -4x_1^2 - x_2^2 + 2.$

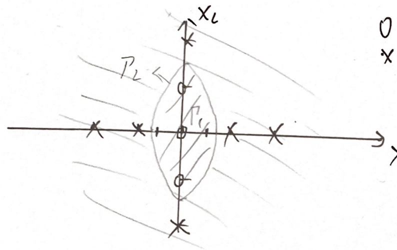
$g(x_1, x_2) \geq 0 \Rightarrow (x_1, x_2) \in S_1$

$g(x_1, x_2) < 0 \Rightarrow (x_1, x_2) \in S_2$

$2 = 4x_1^2 + x_2^2$

$1 = \frac{x_1^2}{\frac{1}{2}} + \frac{x_2^2}{2}$

$\frac{1}{\sqrt{2}}, \sqrt{2}.$



$0 - S_1$

$x - S_2$.