2nd Homework

Chalkiopoulos Georgios | $p3352124$

January 20, 2022

Exercise 1. $y = 3x_1^2 + 4x_2^2 + 5x_3^2 + 7x_1x_2 + x_1x_3 + 4x_2x_3 - 2x_1 - 3x_2 - 5x_3 + \eta$

In this case, y is a nonlinear function of x but a linear function of θ: $\boldsymbol{x} = [x_1, x_2, x_3]^T$ with $\boldsymbol{\theta} = [3, 4, 5, 7, 4, -2, -3, -5]$.

We are looking for a linear transformation, thus we can consider the function transformation:

$$\varphi(\boldsymbol{x}) = \begin{bmatrix} \varphi_1(\boldsymbol{x}) \\ \varphi_2(\boldsymbol{x}) \\ \varphi_3(\boldsymbol{x}) \\ \varphi_4(\boldsymbol{x}) \\ \varphi_5(\boldsymbol{x}) \\ \varphi_6(\boldsymbol{x}) \\ \varphi_7(\boldsymbol{x}) \\ \varphi_8(\boldsymbol{x}) \\ \varphi_9(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ x_1x_2 \\ x_2x_3 \\ x_3x_1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

The problem now becomes linearly separable with:

$$y = 3\varphi_1(\boldsymbol{x}) + 4\varphi_2(\boldsymbol{x}) + 5\varphi_3(\boldsymbol{x}) + 7\varphi_4(\boldsymbol{x}) + \varphi_5(\boldsymbol{x}) + 4\varphi_6(\boldsymbol{x}) - 2\varphi_7(\boldsymbol{x}) - 3\varphi_8(\boldsymbol{x}) - 5\varphi_9(\boldsymbol{x}) + \eta$$

The dimension of the original is : $x : R^3 \rightarrow R^2$ and the transformed is $\varphi : R^9 \rightarrow R^2$.

Exercise 2. $x_1^2 + 3x_2^2 + 6x_3^2 + x_1x_2 + x_2x_3 > (<) 3 \rightarrow \mathbf{x} \in \omega_1(\omega_2)$

The problem is non linear in this form with $\boldsymbol{x} = [x_1, x_2, x_3]^T$. We define:

$$\varphi(\boldsymbol{x}) = \begin{bmatrix} \varphi_1(\boldsymbol{x}) \\ \varphi_2(\boldsymbol{x}) \\ \varphi_3(\boldsymbol{x}) \\ \varphi_4(\boldsymbol{x}) \\ \varphi_5(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ x_1x_2 \\ x_2x_3 \end{bmatrix}$$

The problem becomes linearly separable:

$$\varphi_1(\boldsymbol{x}) + 3\varphi_2(\boldsymbol{x}) + 6\varphi_3(\boldsymbol{x}) + \varphi_4(\boldsymbol{x}) + \varphi_5(\boldsymbol{x}) > (<) 3 \rightarrow \mathbf{x} \in \omega_1(\omega_2)$$

The dimension of the original is : $x : R^3 \rightarrow R^2$ and the transformed is $\varphi : R^5 \rightarrow R^2$.

Exercise 3. Consider the exclusive OR (XOR) classification problem, where the points (0,0) and (1,1) are assigned to class 1 and the points (0,1) and (1,0) are assigned to class +1.

(a) Draw the points on the paper and prove that the classification problem is not linearly separable.

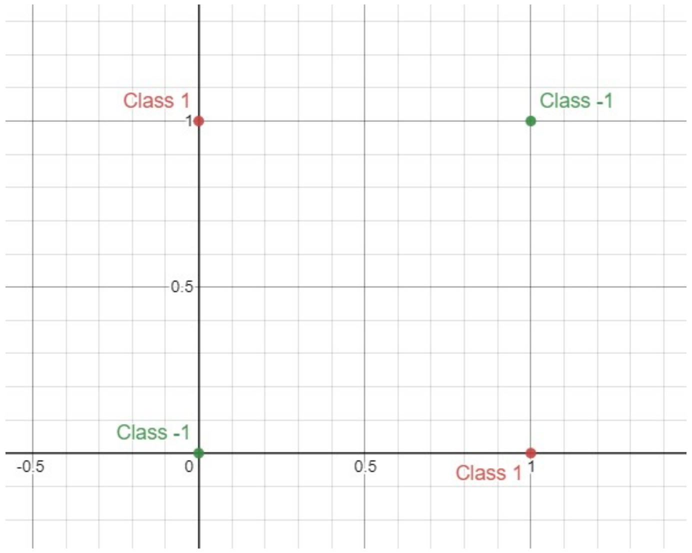


Figure 1: XOR classification points.

We assume that the problem is linearly separable, that is, there exists a hyperplane (H) $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$ that leaves the points from class +1 (resp. -1) on its positive (resp. negative) side. We then need to have:

$$\begin{cases} (0,0) : \theta_0 > 0 \\ (0,1) : \theta_0 + \theta_2 < 0 \\ (1,0) : \theta_0 + \theta_1 < 0 \\ (1,1) : \theta_0 + \theta_1 + \theta_2 > 0 \end{cases} \Rightarrow \begin{cases} (0,0) : \theta_0 > 0 \\ (0,1) : \theta_0 + \theta_2 < 0 \\ (1,0) : \theta_0 + \theta_1 < 0 \\ (1,1) : (\theta_0 + \theta_1) + (\theta_0 + \theta_2) > \theta_0 \end{cases} \tag{3.1}$$

Equation (3.1) cannot be satisfied for any θ since the first part of the last inequality means that θ_0 should be smaller than zero, but from the first inequality we have that θ_0 is greater than zero.

(b) Propose a transformation $\varphi(\cdot)$ that maps the above points to a new space, where the XOR classification problem becomes linearly separable. We could define the following model:

$$abs(x_1 - x_2) > (<)0.5 \rightarrow \boldsymbol{x} \in +1(-1)$$

Exercise 4. Consider a two-class one-dimensional classification problem where the points -3,-1,0,2 belong to class -1 and the points -8,-5,4,7 belong to class +1. Propose a transformation $\varphi(\cdot)$ that maps the above points to a new one-dimensional space, where the classification problem becomes linearly separable.

We first plot the points in a graph, which would make things easier.

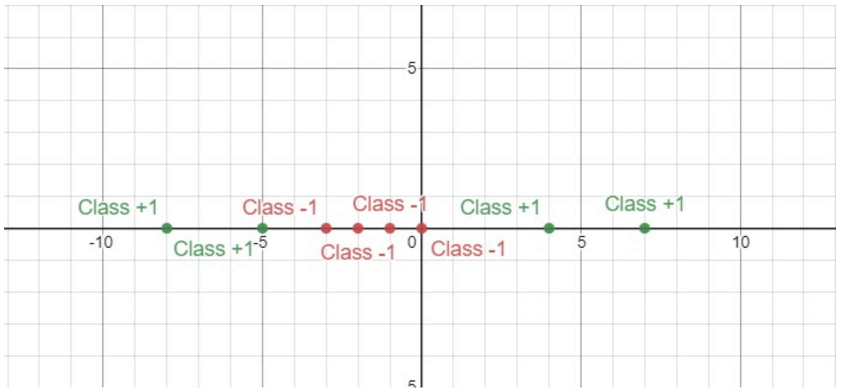


Figure 2: Original points.

In order to separate the points we would need a non-linear separation like $-0.5 + 0.3x^2 > (<)0 \rightarrow \boldsymbol{x} \in +1(-1)$ as shown in figure 3.

We can therefore consider the transformation of $\varphi(\cdot)$:

$$\varphi(x) = |x|$$

The points now become as shown below and can be linearly separated by:

$$\varphi(x) - 3.5 > (<)0 \rightarrow x \in +1(-1)$$

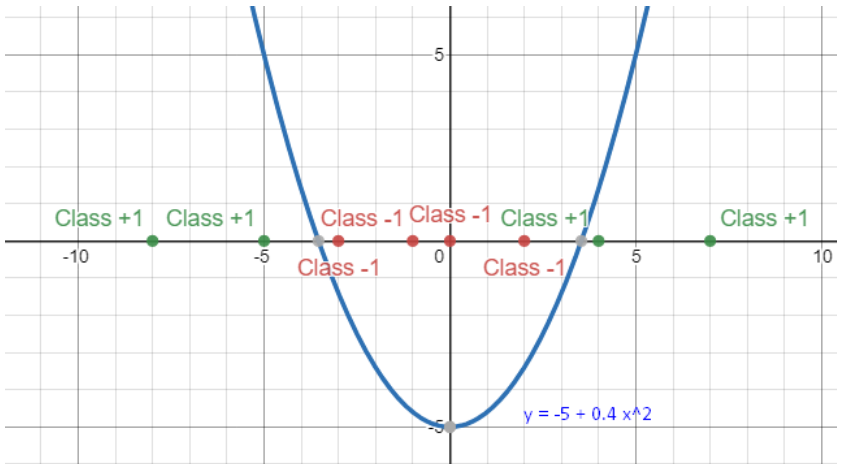


Figure 3: Original points with separation line.

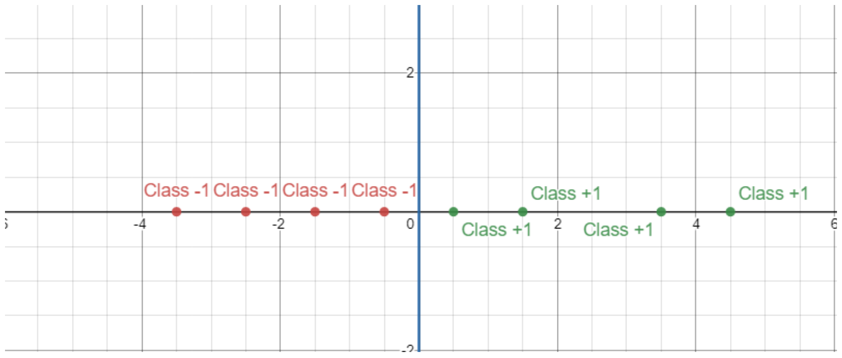


Figure 4: Transformed points with separation line.

Exercise 5. Consider a two-class, two-dimensional classification problem, where the data points $[1,1]^T[1,2]^T$, $[2,1]^T$ belong to class +1, while the data points $[1,1]^T$, $[1,2]^T$, $[2,1]^T$ belong to class -1.

(a) Determine the hyperplane (line) that separates the data from the two classes, utilizing the Least Squares criterion. We define:

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & -2 \\ 1 & -2 & -1 \end{bmatrix} \; and \quad y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

It is: $X^TX = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 12 & 10 \\ 0 & 10 & 12 \end{bmatrix}$ and $(X^TX)^{-1} = \begin{bmatrix} 1/6 & 0 & 0 \\ 0 & 3/11 & -5/22 \\ 0 & -5/22 & 3/11 \end{bmatrix}$.

Also $X^Ty = \begin{bmatrix} 0 \\ 8 \\ 8 \end{bmatrix}$.

Thus $\boldsymbol{\theta} = (X^TX)^{-1}X^Ty = \begin{bmatrix} 1/6 & 0 & 0 \\ 0 & 3/11 & -5/22 \\ 0 & -5/22 & 3/11 \end{bmatrix} \begin{bmatrix} 0 \\ 8 \\ 8 \end{bmatrix} = \begin{bmatrix} 0 \\ 4/11 \\ 4/11 \end{bmatrix}$

Thus, the Least Squares line is

$$\frac{4}{11}x_1 + \frac{4}{11}x_2 = 0$$

(b) Draw on the paper the data points from the classes along with the line determined in (a).

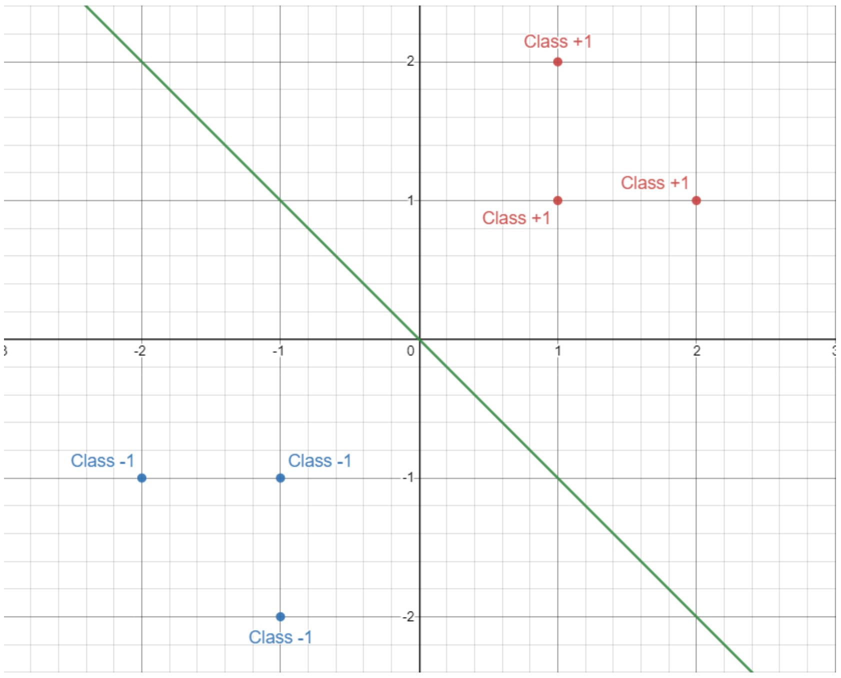


Figure 5: Data points with Least Squares line.

(c) Is this the unique line that separates the data from the two classes? Is this line the only one that results from the minimization of the least squares criterion?

This is not the only line that separates the data. There are infinite lines that could separate the data points. That said, for this specific problem (with classes +1, -1) this is the only line that results from minimization using the least squares criterion.

Exercise 6

6.a

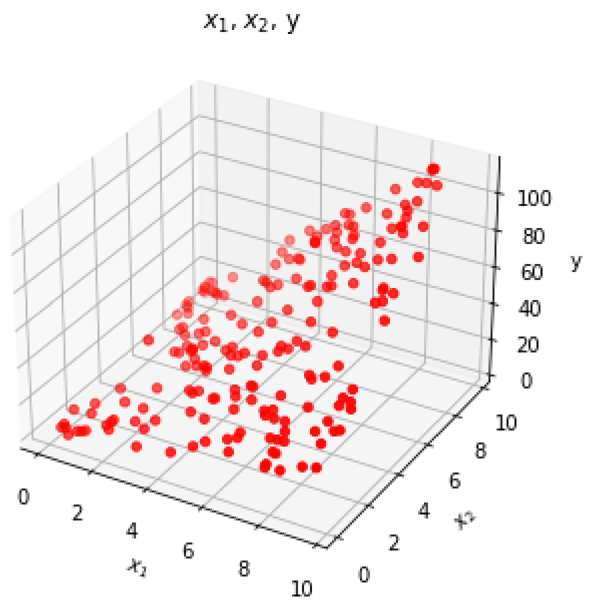
Generate the set.

```
In [ ]:
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import pandas as pd

# for creating a responsive plot
%matplotlib inline
```

```
In [ ]:
# Construct X matrix [1, x1, x2, x1*x2]
X = np.ones((200, 1))
X = np.append(X, np.random.uniform(low=0,high=10,size=(200,2)), axis=1)
X = np.append(X, (X[:,1] * X[:,2]).reshape(-1,1), axis=1)

# define theta
theta = np.array([[3, 2, 1, 1]])

# define normal error
n = np.random.normal(0,np.sqrt(0.05),len(X))

# Define y using only x1, x2
y = theta.dot(X.T) + n

#prin X and y
np.append(X[:,1:], y.reshape(-1,1), axis=1)[:5]
```

```
Out[ ]:
array([[ 3.17902621,  1.23782135,  3.93506652, 14.41493732],
       [ 3.93239446,  4.34573743, 17.08915378, 32.01851452],
       [ 9.65962238,  2.0651446 , 19.94851701, 44.31936188],
       [ 3.03617522,  0.79596378,  2.4166855 , 12.64137055],
       [ 1.58726582,  1.98523876,  3.15110162, 11.09764923]])
```

```
In [ ]:
#plot X data
fig = plt.figure(figsize=(7,5))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:,1],X[:,2],y,c='r',marker='o')
ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_zlabel('y')
ax.set_title('$x_1$, $x_2$, y')
plt.show()
```



In [ ]:
```python
# For the rest we assume that we only know xi1, xi2 and yi
X_r = X[:,0:3]
```

6.b

Adopting the linear model assumption in the original space and the LS criterion, estimate the parameters of the model

- Using the least squares method, assuming that the relation is linear, we can calculate:

In [ ]:
```python
Xx_inv = np.linalg.inv(X_r.T.dot(X_r))

Xy = X_r.T.dot(y.T)

# Finally
theta_r = Xx_inv.dot(Xy)
print(theta_r.flatten())
```

```
[-20.81711347   6.71265961   5.99903766]
```

6.c

For each one of the 200 data points x_i of X, determine the associated estimate $\hat{y}i$ provided from the model estimated in (b) and compute the MSE

In [ ]:
```python
y_hat = theta_r.T.dot(X_r.T)

MSE = np.power((y-y_hat), 2).mean()
print(f"The MSE is {MSE}")
```

```
The MSE is 82.32176609421839
```

6.d

- We will now use our original data since we need x1, x2 and x1*x2. Let's see the first 5 rows

```
In [ ]: X[:5, 1:]
```

```
Out[ ]: array([[ 3.17902621,  1.23782135,  3.93506652],
               [ 3.93239446,  4.34573743, 17.08915378],
               [ 9.65962238,  2.0651446 , 19.94851701],
               [ 3.03617522,  0.79596378,  2.4166855 ],
               [ 1.58726582,  1.98523876,  3.15110162]])
```

6.e

Adopting the linear model assumption in the original space and the LS criterion, estimate the parameters of the model

- Using the least squares method, assuming that the relation is linear, we can calculate:

```
In [ ]: Xx_inv_3 = np.linalg.inv(X.T.dot(X))

        Xy_3 = X.T.dot(y.T)

        # Finally
        theta_r_3 = Xx_inv_3.dot(Xy_3)
        print(theta_r_3.flatten())
```

```
[2.89605155 2.01815    1.01742978 0.99736414]
```

6.f

For each one of the 200 data points xi of X, determine the associated estimate $\hat{y}i$ provided from the model estimated in (b) and compute the MSE

```
In [ ]: y_hat_3 = theta_r_3.T.dot(X.T)

        MSE = np.power((y-y_hat_3), 2).mean()
        print(f"The MSE is {MSE}")
```

```
The MSE is 0.04412455677594366
```

6.g

Having calculated y as a non-linear function of $x_1$ $x_2$ we initially found estimates of $\theta_0, \theta_1, \theta_2$ using the LS method. The results aren't promising, since y is a non linear function and we tried to estimate it using a linear function $f_\theta(x)$, thus the very high MSE. In 6.f we applied the transformation, in which our data matched the original ones, and the estimate of θ are very close to the calculated ones.

Exercise 7

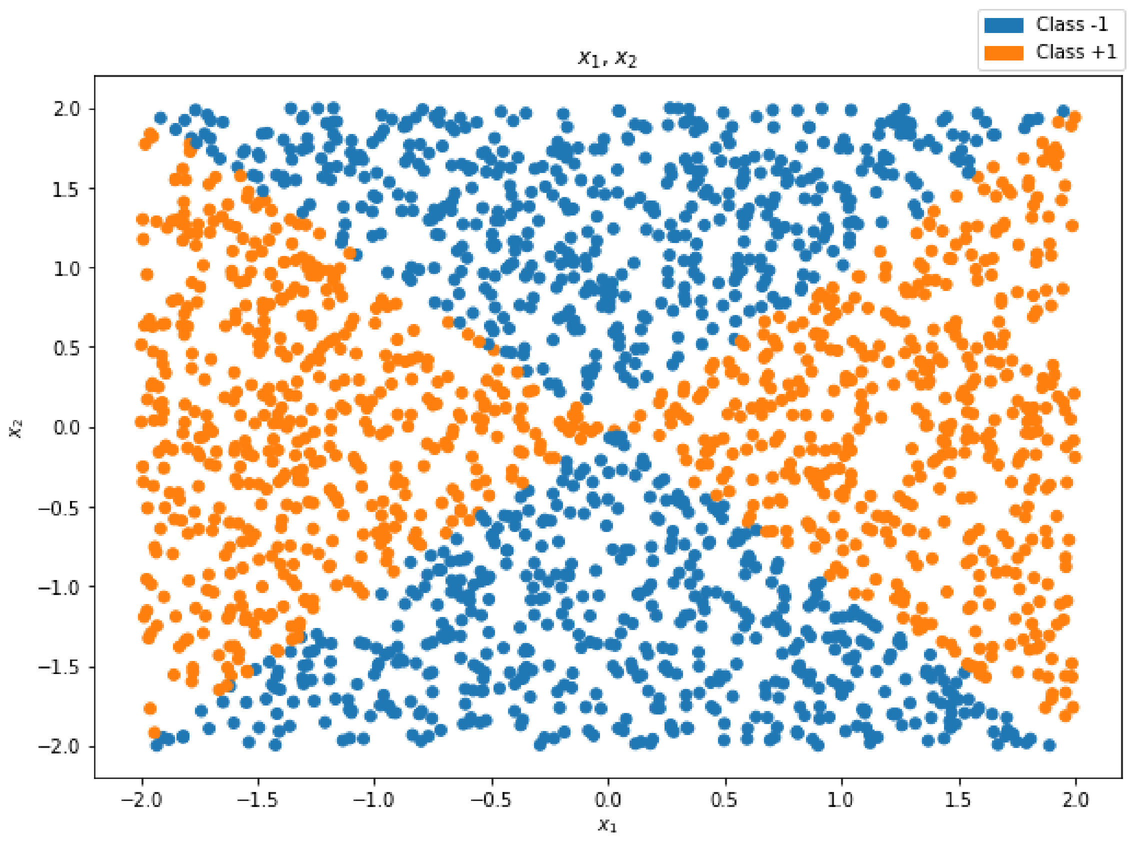
7.a

Generate the set and plot the points.

In [ ]:
```
# Construct X matrix [1, x1, x2, x1*x2]
x_ones = np.ones((2000, 1))
x1 = np.random.uniform(low=-2,high=2,size=(2000,1))
x2 = np.random.uniform(low=-2,high=2,size=(2000,1))
X = np.concatenate((x_ones, x1, x2), axis=1)

# Classify the points based on x1^2 - x2^2 = 0, +1 if >= 0 else -1
y = np.where(np.power(x1,2) - np.power(x2,2) >= 0, +1, -1)

# print X and y
np.append(X[:,1:], y.reshape(-1,1), axis=1)[:5]
```

Out[ ]:
```
array([[ 0.7610215 , -0.2912994 ,  1.        ],
       [-0.94011821, -1.75415967, -1.        ],
       [-0.73488283,  1.20530662, -1.        ],
       [-0.39812191, -0.62127847, -1.        ],
       [ 1.56338613,  0.08112157,  1.        ]])
```

In [ ]:
```
# define color pattern
color = pd.Series(y.flatten()).apply(lambda x: 'C0' if x == -1 else 'C1')
# Create patches for the legend
patches = [ mpatches.Patch(color = 'C0', label = 'Class -1'), mpatches.Patch(c


#plot X data
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot(111)
ax.scatter(X[:,1],X[:,2],c=color,marker='o')
ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_title('$x_1$, $x_2$')
_ = ax.legend(handles = patches, \
            loc = 'upper left', bbox_to_anchor=(0.85, 0, 0, 1.11))
```



7.b

Apply the transformation

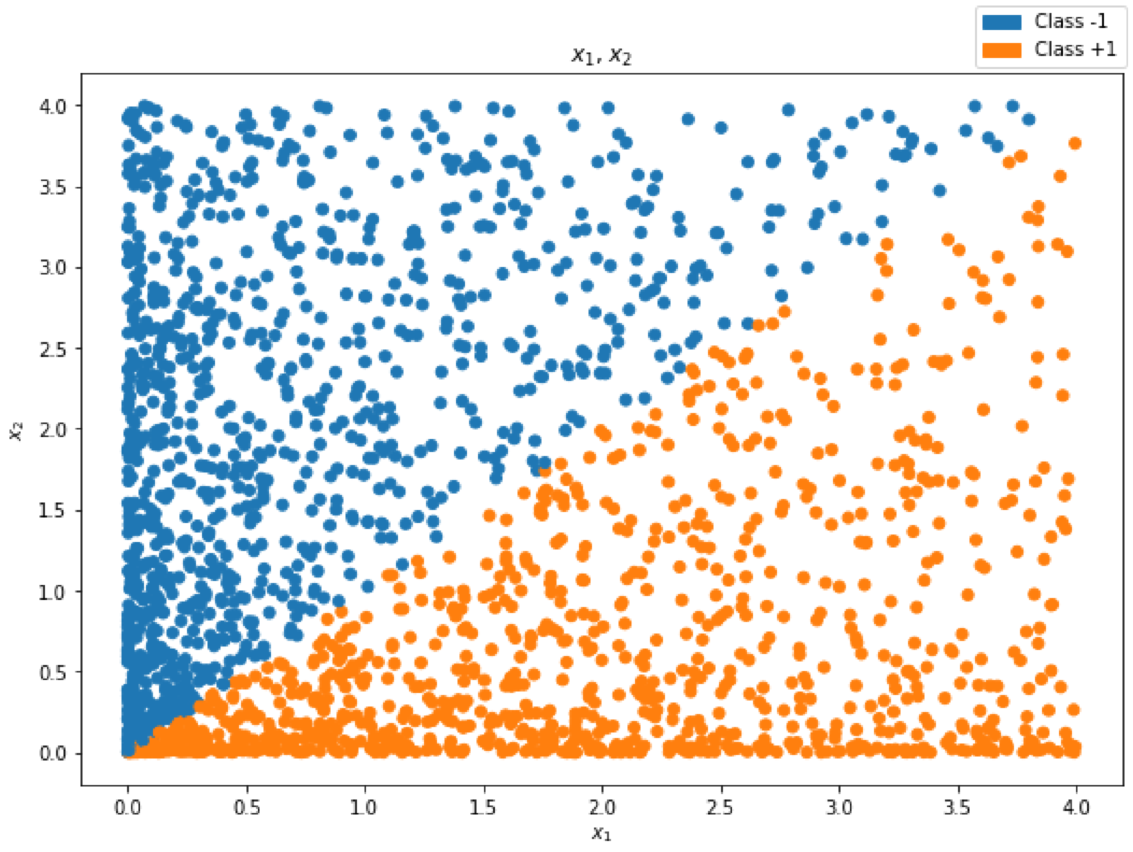
In [ ]:
```
X_new = np.power(X[:,1:], 2)

# print X and y
np.append(X_new, y, axis=1)[:5]
```

Out[ ]:
```
array([[ 0.57915372,  0.08485534,  1.        ],
       [ 0.88382225,  3.07707614, -1.        ],
       [ 0.54005277,  1.45276404, -1.        ],
       [ 0.15850105,  0.38598694, -1.        ],
       [ 2.44417621,  0.00658071,  1.        ]])
```

7.c

Plot and comment

In [ ]:
```
#plot X data
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot(111)
ax.scatter(X_new[:,0],X_new[:,1],c=color,marker='o')
ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_title('$x_1$, $x_2$')
_ = ax.legend(handles = patches, \
            loc = 'upper left', bbox_to_anchor=(0.85, 0, 0, 1.11))
```



In the first case (before the transformation) our data was not linearly separable, which is obvious looking at Figure 2. When we applied the transformation our data became linearly separable. This happened because in the first case y was a function of x_1^2 and x_2^2 (not linear) while in the second case it it a function of $\phi_1(x)_1$ and $\phi_2(x)$ (which is linear).

7.d

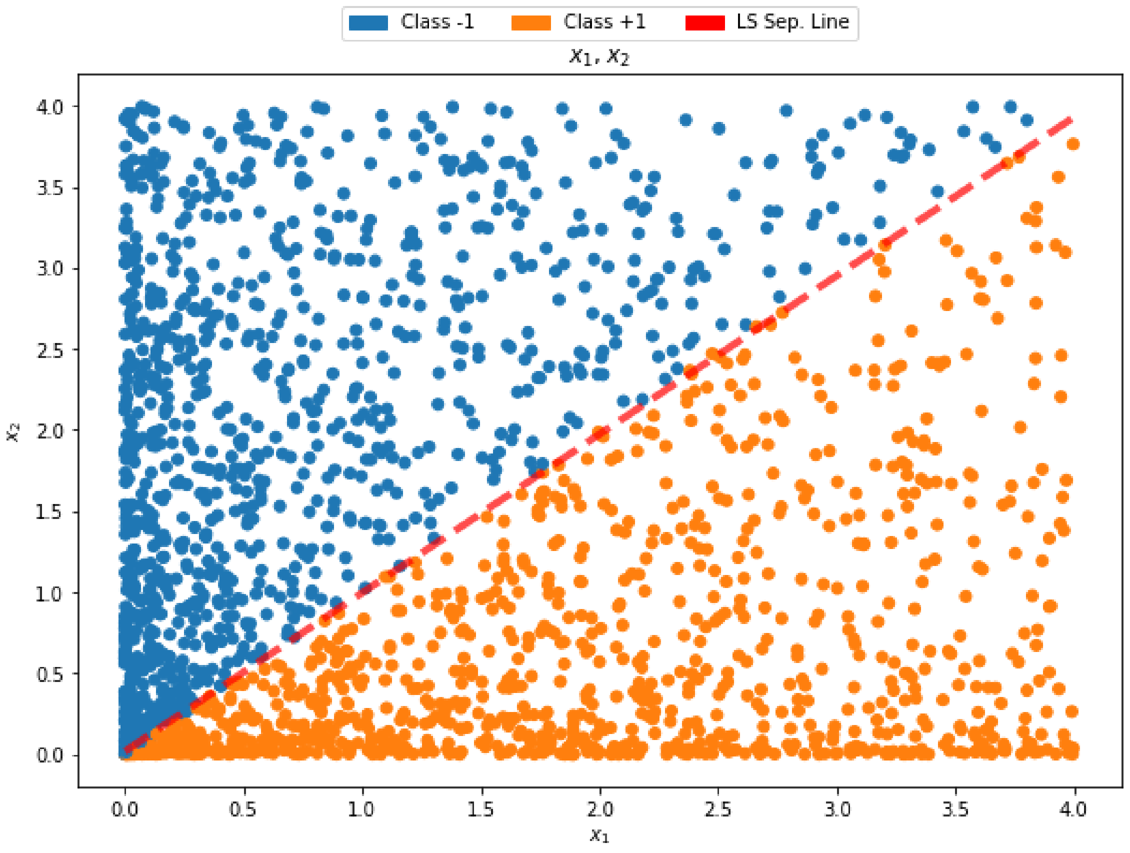
Estimate the parameters

In [ ]:
```
# add ones to the transformed table for LS
X_new = np.append(x_ones, X_new, axis = 1)

# Performs LS
Xx_inv = np.linalg.inv(X_new.T.dot(X_new))
Xy = X_new.T.dot(y)

# Calculate and print coeff
theta = Xx_inv.dot(Xy)
print(theta)
```

```
[[ 0.00854722]
 [ 0.46595532]
 [-0.47635879]]
```

In [ ]:
```python
# Create points to plot
lim = np.array([0,1,2,3,4])
line = - theta[0]/theta[2] - theta[1]/theta[2]*lim

# add LS Sep.line patch
patches.append(mpatches.Patch(color = 'r', label = 'LS Sep. Line'))
```

In [ ]:
```python
#plot X data
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot(111)
ax.scatter(X_new[:,1],X_new[:,2],c=color,marker='o')
ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_title('$x_1$, $x_2$')
ax.plot(lim, line, linewidth=4, linestyle='--', color='r', alpha=0.7)
# ax.legend(bbox_to_anchor=(1.01, 0, 0, 1.11))
_ = ax.legend(handles = patches, \
            loc = 'upper center', bbox_to_anchor=(0.5, 0, 0, 1.11), ncol=3)
```



Exercise 8

8.a

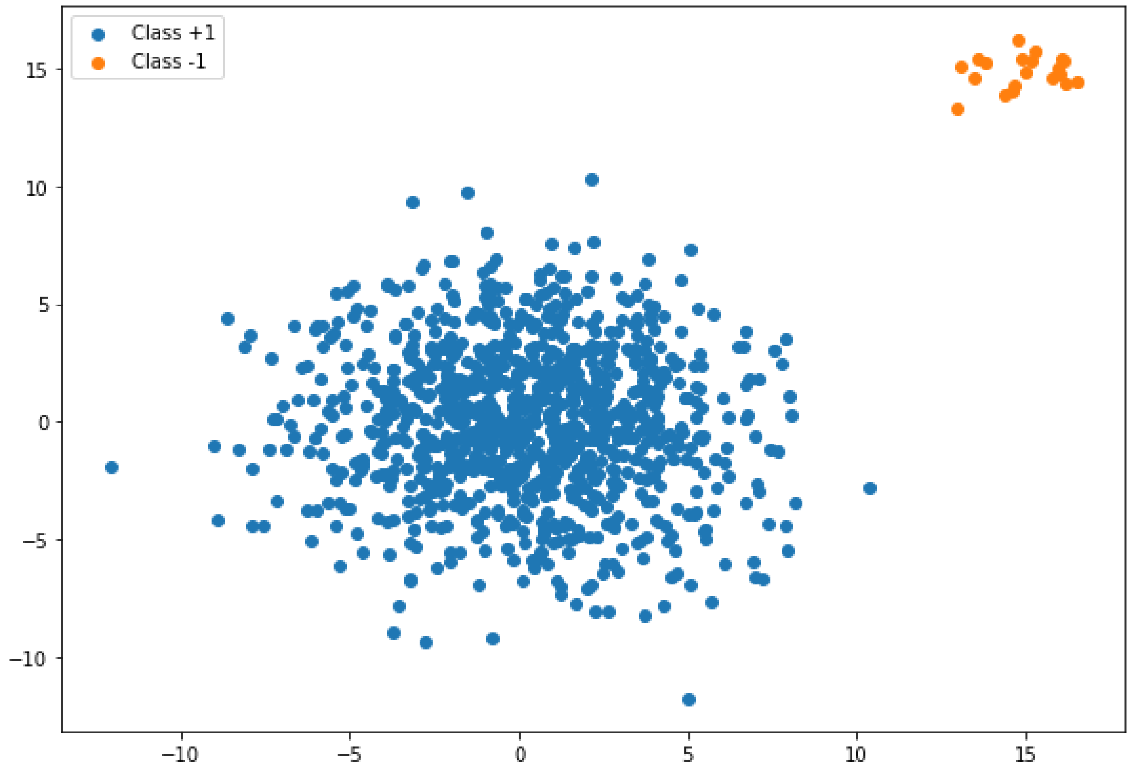
Generate the set and plot the points.

In [ ]:

```
# Generate data
mean_1 = [0,0]
cov_1 = [[10,0],[0,10]]
mean_2 = [15,15]
cov_2 = [[1,0],[0,1]]

x1, x2 = np.random.multivariate_normal(mean_1,cov_1,1000), np.random.multivari

X = np.concatenate((x1, x2))
y = np.append(np.full((1000,1), 1), np.full((20,1), -1))
```

In [ ]:

```
#plot X data
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot(111)
ax.scatter(X[:1000,0],X[:1000,1],c='C0',marker='o', label='Class +1')
ax.scatter(X[1000:,0],X[1000:,1],c='C1',marker='o', label='Class -1')
_ = ax.legend()
```



8.b

Determine the sparating line.

```
In [ ]: # add ones to the transformed table for LS
        X_new = np.append(np.ones((len(X),1)), X, axis = 1)

        # Performs LS
        Xx_inv = np.linalg.inv(X_new.T.dot(X_new))
        Xy = X_new.T.dot(y)

        # Calculate and print coeff
        theta = Xx_inv.dot(Xy)
        print(theta)
```

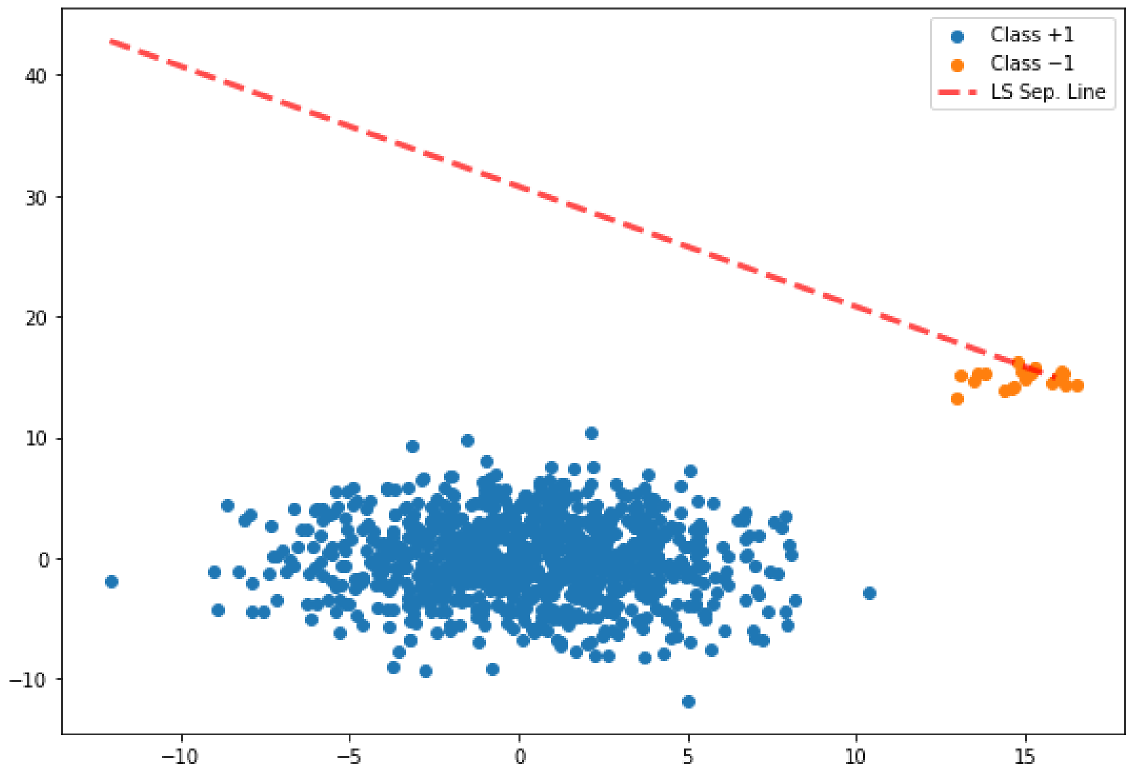
```
[ 0.9831993  -0.03183185 -0.03196898]
```

8.c

Plot the data points with the line

```
In [ ]: # Create points to plot
        lim = np.arange(np.amin(X[:]), np.amax(X[:]))
        line = - theta[0]/theta[2] - theta[1]/theta[2]*lim
```

```
In [ ]: #plot X data
        fig = plt.figure(figsize=(10,7))
        ax = fig.add_subplot(111)
        ax.scatter(X[:1000,0],X[:1000,1],c='C0',marker='o', label='Class +1')
        ax.scatter(X[1000:,0],X[1000:,1],c='C1',marker='o', label='Class -1')
        ax.plot(lim, line, linewidth=3, linestyle='--', color='r', label='LS Sep. Line
        _ = ax.legend()
```



8.d

Comment on the results

- As we can see the separating line does a bad job of separating the two classes. As we have seen in the lectures (Lecture 2 slides) this is caused by the imbalance in the number of points belonging to each class. Altought the points belonging in Class -1 have a bigger $J(\theta)$ the fact that there are much less points in this class, than those belonging in Class +1, means that they contribute less in the overall Cost Function. If we had even more points in Class +1, the line would be further away from the points. In that case the LS criterion would results in a larger value for "Class -1" points, but a smaller one overall.

```
In [ ]:
```