Machine Learning and Computational Statistics
HW 2 Giagtos Stylianos
Exercise 1.

Consider the following nonlinear model:

$y = 3x_1^2 + 4x_2^2 + 5x_3^2 + 7x_1x_2 + x_1x_3 + 4x_2x_3 - 2x_1 - 3x_2 - 5x_3 + 4$

Define a suitable function φ that transforms the problem to a space where the problem of estimating the model becomes linear. What is the dimension of the original and the transformed spaces?

The task is to define a function $\phi(x)$ that maps the original vector $x = [x_1, x_2, x_3]$ into higher dimensional space by capturing nonlinear terms of the model into linear ones.

The transformation function is:

$\phi[x] = [x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3, x_1, x_2, x_3]$

Therefore, the original model can be transformed into linear one in the new 9-dimensional space.

So, $y = 3\cdot\phi_1(x) + 4\phi_2(x) + 5\varphi_3(x) + 7\varphi_4(x) + 1\cdot\varphi_5(x) + 4\varphi_6(x) - 2\varphi_7(x) - 3\varphi_8(x) - 5\varphi_9(x) + 4.$

The original space consists of 3 dimensions $x = [x_1, x_2, x_3]$ and the transformed space consists of 9 dimensions.

$\phi(x) = [x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3, x_1, x_2, x_3]$

Exercise 2.

Consider the following two-class nonlinear classification task.

$x = [x_1, x_2, x_3]^T : x_1^2 + 3x_2^2 + 6x_3^2 + x_1x_2 + x_2x_3 > (<)\ 3 \rightarrow x \in \omega_1(\omega_2)$

Define a suitable function ϕ that transforms the problem to a space where the problem of estimating the border of the two classes becomes linear. What is the dimension of the original and the transformed spaces?

Again by viewing the problem from the same scope we need a transformation that includes all the nonlinear terms and maps the features into a higher-dimensional space.

The transformation is: $\phi[x] = [x_1^2, x_2^2, x_3^2, x_1x_2, x_2x_3]$

So the boundary now is: $\phi_1 + 3\phi_2 + 6\phi_3 + \phi_4 + \phi_5 = 3$.

with the respective transformations of:

$\phi_1 = x_1^2, \phi_2 = x_2^2, \phi_3 = x_3^2, \phi_4 = x_1x_2, \phi_5 = x_2x_3$

So the problem is now a linear classification problem of function $\phi_1 + 3\phi_2 + 6\phi_3 + \phi_4 + \phi_5 = 3$.

The original space dimensions are 3. with the features vector $x = (x_1, x_2, x_3)$.

The transformed space dimensions are 5 with the features vector $\phi(x) = [x_1^2, x_2^2, x_3^2, x_1x_2, x_2x_3]$.

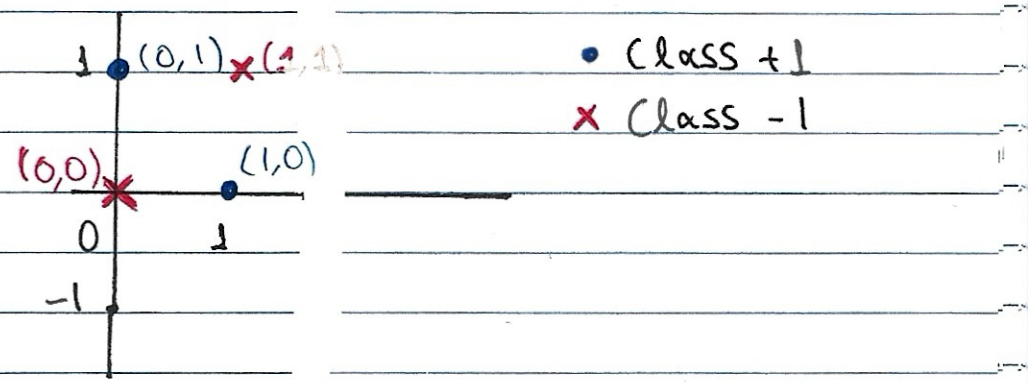
Exercise 3

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 +1

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

b) Propose a transformation $\phi(\cdot)$ that maps the above points to a new space, where the XOR classification problem becomes linearly separable.

a) Here is the plot presenting the XOR problem, where points (0,0) and (1,1) are assigned to class -1. and points (0,1) and (1,0) are assigned to class +1.



By assuming that the problem is linearly separable the equation of the Hyperplane is $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$.

So the points in class +1 satisfy: $\theta_0 + \theta_1 x_1 + \theta_2 x_2 > 0$

and the points in class -1 satisfy: $\theta_0 + \theta_1 x_1 + \theta_2 x_2 < 0$.

- for point (0,0) → Class -1 $\quad \theta_0 + \theta_1(0) + \theta_2(0) = \theta_0 < 0 \quad (1)$
- for point (1,1) → Class -1 $\quad \theta_0 + \theta_1(1) + \theta_2(1) = \theta_0 + \theta_1 + \theta_2 < 0 \quad (2)$
- for point (0,1) → Class +1 $\quad \theta_0 + \theta_1(0) + \theta_2(1) = \theta_0 + \theta_2 > 0 \quad (3)$
- for point (1,0) → Class +1 $\quad \theta_0 + \theta_1(1) + \theta_2(0) = \theta_0 + \theta_1 > 0. \quad (4)$

From inequality (1) we get $\theta_0 < 0$

Respectively, from (2) $\theta_0 + \theta_2 > 0 \Rightarrow \theta_2 > -\theta_0$

Also, from (4) $\theta_0 + \theta_1 > 0 \Rightarrow \theta_1 > -\theta_0$

From inequality (2) $\theta_0 + \theta_1 + \theta_2 < 0$

By substitution of: $\theta_0 + \theta_1 + \theta_2 < 0$

$\theta_1 > -\theta_0$

$\theta_2 > -\theta_0$ (-1)

We get the following inequality.

$\theta_0 - \theta_0 - \theta_0 < 0 \Rightarrow -\theta_0 < 0 \Rightarrow \theta_0 > 0$.

Which is a contradiction to the initial inequality (1) $\theta_0 < 0$ so the initial assumption does not hold.

Therefore the XOR problem is not linearly separable in the two dimentions featurespace. as, it is impossible to find a straight line to separate the two classes without contradiction.

3b) So we have the points and their respective classes

Class -1: (0,0) and (1,1)

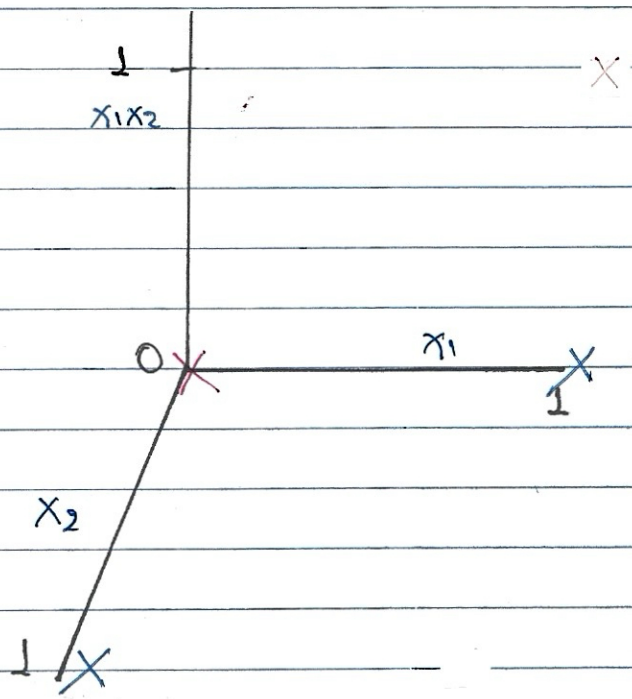
Class +1: (0,1) and (1,0)

Since the points can't be separated linearly a transformation is needed. By adding the feature $x_1 \cdot x_2$ the problem becomes linearly separable in a 3 dimensional space

Class -1: (0,0,0), (1,1,1)

Class +1: (0,1,0), (1,0,0)

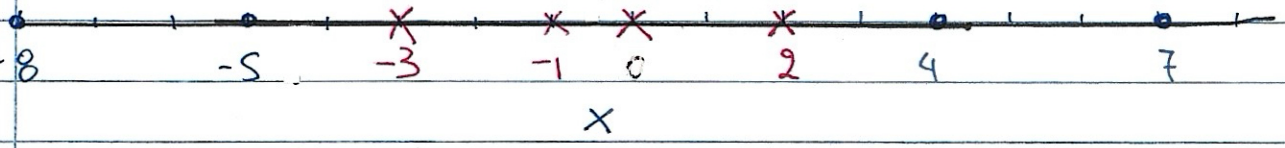
So the used transformation is $\varphi(x_1, x_2) = (x_1, x_2, x_1 \cdot x_2)$



Exercise 4

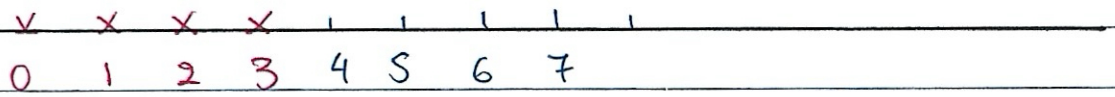
Consider a two-class one-dimensional classification problem where the points $-3, -1, 0, 2$ belong to class -1 and the points $-8, -5, 4, 7$ belong to class $+1$. Propose a transformation $\varphi(\cdot)$ that maps the above points to a new one dimension space where the classification problem becomes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

Initially, we will try to plot the classes on the paper in order to see that it is not separable in a more schematical way. × Class -1 • Class +1

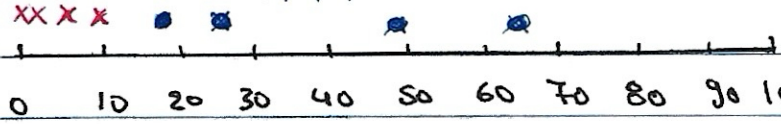


As it seems the points are located in a line, and their order does not help to separate it.
By removing the negative numbers it is noticed that the classes are in a good order in a separable way. By using the transformation of $\varphi(x) = |x|$ we transform all numbers into positive linearly separable ones. So the decision boundary can now be around 3, 4.

× Class -1 • Class +1



Another transformation would also be (x^2), $\varphi(x) = x^2$. which would also transform the data into positive and most significant linearly separable data. in the transformed space. Class -1: 9, 1, 0, 4, Class +1: 64 25 16 49



× Class -1
• Class +1

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.

We define $X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & -2 \\ 1 & -2 & -1 \end{bmatrix}$, $y = \begin{bmatrix} +1 \\ +1 \\ +1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & -1 & -1 & -2 \\ 1 & 2 & 1 & -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & -2 \\ 1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 12 & 10 \\ 0 & 10 & 12 \end{bmatrix}$$

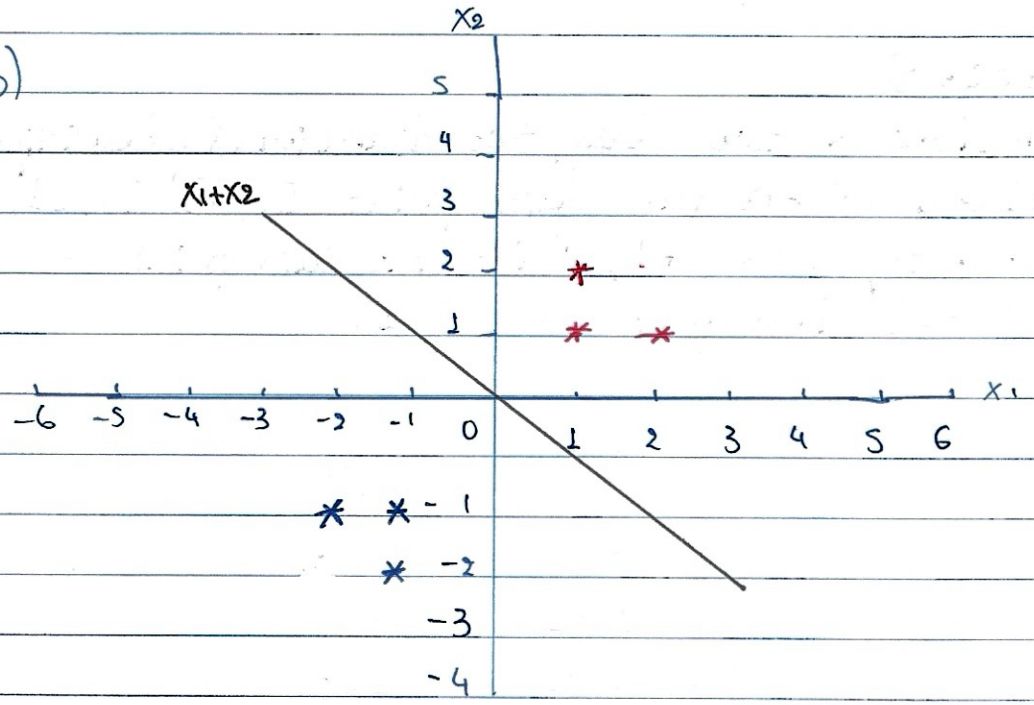
$$(X^T X)^{-1} = \frac{\text{Adj}(X^T X)}{\text{Determinant}} = \frac{1}{264} \begin{bmatrix} 44 & 0 & 0 \\ 0 & 72 & -60 \\ 0 & -60 & 72 \end{bmatrix} = \begin{bmatrix} 0.167 & 0 & 0 \\ 0 & 0.273 & -0.227 \\ 0 & -0.227 & 0.273 \end{bmatrix}$$

$$X^T y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & -1 & -1 & -2 \\ 1 & 2 & 1 & -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\theta = (X^T \cdot X)^{-1} X^T y = \begin{bmatrix} 0.167 & 0 & 0 \\ 0 & 0.273 & -0.227 \\ 0 & -0.227 & 0.273 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.36 \\ 0.36 \end{bmatrix}$$

So the Least Square Line is: $0.36x_1 + 0.36x_2 = 0$

b)



c) The line X_1+X_2 can't be unique as in the 2D space there are many other lines with different slope that could still separate the data into two classes.

Exercise 6

Let $x = [x_1, \dots, x_l]^T$ be an l-dimensional random vector with mean vector $\mu = [\mu_1, \dots, \mu_l]^T$ and let $Cov(x) = E[(x-\mu)\cdot(x-\mu)^T]$ and $R_x = E[x \cdot x^T]$ be the corresponding covariance and correlation matrices respectively. Prove that $R_x = cov(x) + \mu \cdot \mu^T$

$$E[(x-\mu)\cdot(x-\mu)^T] = E[xx^T - x\mu^T - \mu x^T + \mu\mu^T]$$
$$= E[xx^T] - E[x]\mu^T - \mu E[x^T] + \mu\mu^T$$

We have that $E[x] = \mu$, this is

$$E[xx^T] - \mu\mu^T.$$

So $Cov(x) = R_x - \mu\mu^T \Rightarrow R_x = Cov(x) + \mu\mu^T$

Exercise 7

a) Prove that the mean and the variance of a random variable x that follows the Bernoulli distribution $Ber(x|p)$ $(0<p<1)$ are $E[x] = p$ and $\sigma_x^2 = p(1-p)$ respectively.

The PMF of a Bernoulli random variable is $P(x=1) = p$ and $P(x=0) = 1-p$.

$E[x] = 1 \cdot p + 0 \cdot (1-p) = p$.

$Var(x) = E(x^2) - (E(x))^2$ (1) But x can only be 0 or 1 and $x^2 = x$, then $E(x^2) = E[x] = p$ (2).

From (1) and (2): $Var(x) = p - p^2 = p(1-p)$

b) Prove that the mean of the random variable x that follows the binomial distribution $Bin(x|n,p)$ $(0<p<1)$ is $E[x]=np$.

The PMF of the binomial distribution $x \sim B(n,p)$ is

$$P(x=k) = \binom{n}{k} p^k (1-p)^{n-k}$$

The weighted average of all possible values of x, weighted by their probabilities is

$$E[x] = \sum_{k=0}^{n} k \binom{n}{k} p^k (1-p)^{n-k}$$

$k\binom{n}{k} = n\binom{n-1}{k-1}$, as $k\binom{n}{k}$ is equivalent to k objects from n objects so $E[x]$ is also equivalent to

$$E[x] = n \sum_{k=1}^{n} \binom{n-1}{k-1} p^k (1-p)^{n-k}$$

Let $k' = k-1$, k' runs from 0 to $n-1$ then we have

$$E[x] = n \sum_{k'=0}^{n-1} \binom{n-1}{k'} p^{k'+1} (1-p)^{n-k'-1} \Leftrightarrow$$

$$E[x] = np \sum_{k'=0}^{n-1} \binom{n-1}{k'} p^{k'} (1-p)^{n-1-k'}$$

The underlined red part resembles the binomial expansion $(p+(1-p))^{n-1}$ which is equal to $1^{n-1} = 1$.

So we have $E(x) = np \times 1 = np$.
This proves the expected value of $x \sim Bin(n,p)$ is $E[x] = np$.

Exercise 8

(a) Data Generation

Generate a set

$$X = \{(y_i, \mathbf{x}_i), \mathbf{x}_i = [x_{i1}, x_{i2}]^T \in \mathbb{R}^2, y_i \in \mathbb{R}, i = 1, \ldots, 200\}$$

from the model

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \eta$$

where η is an i.i.d. normal zero mean noise with variance 0.05. Use the parameters $\theta_0 = 3, \theta_1 = 2, \theta_2 = 1, \theta_3 = 1$ (adopt the strategy given in the example on the second slide of the second lecture). In the sequel, pretend that you do not know the model that generates the data. All you have at your disposal is the data set X.

In [1]:
```
import numpy as np
import pandas as pd

# Given parameters for the model
theta_0 = 3
theta_1 = 2
theta_2 = 1
theta_3 = 1
n = 200  # Number of data points
noise_variance = 0.05

# Generate random x1 and x2 values from a normal distribution
np.random.seed(23)  # Set seed for reproducibility
x1 = np.random.randn(n)  # 200 samples from a standard normal distribution
x2 = np.random.randn(n)

# Generate noise from a normal distribution
eta = np.random.normal(0, np.sqrt(noise_variance), n)

# Compute the dependent variable y based on the given model
y = theta_0 + theta_1 * x1 + theta_2 * x2 + theta_3 * x1 * x2 + eta

# Combine the data into a DataFrame (y, [x1, x2])
data = pd.DataFrame({'x1': x1, 'x2': x2, 'y': y})

# Show the first few rows of the dataset
data.head()
```

Out[1]:

	x1	x2	y
0	0.666988	0.572950	5.370828
1	0.025813	-0.652938	2.537379
2	-0.777619	1.690587	1.471532
3	0.948634	0.820629	6.964190
4	0.701672	0.109570	4.659354

(b) Parameter Estimation in the Original Space

Adopting the linear model assumption in the original space (that is, assuming that

$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2$, estimate the parameters of the model $(\theta_0, \theta_1, \theta_2)$ that minimize the sum of error squares criterion.

The given model is: y=θ0+θ1χ1+θ2χ2 and it can be written in a matrix form of y=XΘ+η where X is the features matrix and θ is the parameters vector. θ0 θ1 and θ2 will be estimated using least squares method by solving this equation

$\boldsymbol{\theta} = (X^T X)^{-1} X^T \mathbf{y}$

In [2]:
```
# Adding a column of ones for the intercept term (theta_0)
X = np.column_stack((np.ones(n), x1, x2))

# Performing Ordinary Least Squares (OLS) to estimate theta
theta_hat = np.linalg.inv(X.T @ X) @ X.T @ y

theta_hat
```

Out[2]:
```
array([2.95406672, 1.74273132, 1.05513799])
```

Parameters are: Θ0=2.95, θ1=1.74, θ2=1.05

(c) Mean Squared Error Calculation

For each of the 200 data points $\mathbf{x}_i$ of X, determine the associated estimate $\hat{y}_i$ provided by the model estimated in (b) and compute the mean squared error (MSE):

$MSE = \frac{1}{200} \sum_{i=1}^{200} (y_i - \hat{y}_i)^2$

In [3]:
```
# Compute the predicted y values using the estimated parameters
y_hat_b = X @ theta_hat

# Compute the MSE for the model estimated in (b)
mse_b = np.mean((y - y_hat_b) ** 2)

mse_b
```

Out[3]:
```
0.698987595288398
```

MSE for model in B) is 0.698

(d) Data Transformation

Apply the transformation

$$\phi(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ \phi_3(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1 \cdot x_2 \end{bmatrix}$$

on all $\mathbf{x}_i$ of X. Denote by $\mathbf{x}'_i \in \mathbb{R}^3$ the image of $\mathbf{x}_i$ (that is, $\mathbf{x}'_i = \phi(\mathbf{x}_i)$), and form a new data set

$$X' = \{(y_i, \mathbf{x}'_i), \mathbf{x}'_i \in \mathbb{R}^3, y_i \in \mathbb{R}, i = 1, \ldots, 200\}$$

In [4]:
```
# Apply the transformation to include the interaction term
x1_x2 = x1 * x2
X_prime = np.column_stack((np.ones(n), x1, x2, x1_x2))
```

(e) Parameter Estimation in the Transformed Space

Adopting the linear model assumption in the transformed space and the sum of error squares criterion, estimate the parameters of the model that minimize this criterion.

In [5]:
```
# Perform OLS to estimate theta in the transformed space
theta_prime_hat = np.linalg.inv(X_prime.T @ X_prime) @ X_prime.T @ y

theta_prime_hat
```

Out[5]:
```
array([2.9667611 , 1.99845192, 0.98927105, 0.99921008])
```

The new θ values are θ0=2.97, θ1=1.99, θ2=0.98, θ3=0.99

(f) Mean Squared Error in the Transformed Space

For each of the 200 data points $\mathbf{x}_i$ of X, determine the associated estimate $\hat{y}_i$ provided by the model estimated in (e) and compute the mean squared error (MSE):

$$MSE = \frac{1}{200} \sum_{i=1}^{200} (y_i - \hat{y}_i)^2$$

In [6]:
```
# Compute the predicted y values using the estimated parameters in the transformed
y_hat_prime = X_prime @ theta_prime_hat

# Compute the MSE for the model estimated in (e)
mse_prime = np.mean((y - y_hat_prime) ** 2)

mse_prime
```

Out[6]:
```
0.046107680534206 83
```

MSE for model in e) is 0.046

(g) Comments on Results

Comment on the results obtained in (c) and (f).

The MSE (Mean Squared Error) of the model that includes the interaction term (0.046) is much lower than the MSE of the simpler linear model (0.699). This shows that the model with the interaction term fits the data much better. Since the true data includes an interaction between $x1$ and x2, the model with this term is able to capture the real relationship more accurately. As a result, adding the interaction term gives a more precise estimate of the outcome, y.

Exercise 9

(a) Data Generation

Generate a set $X = \{(y_i, \mathbf{x}_i), \mathbf{x}_i \in \mathbb{R}^2, y_i \in \{-1, +1\}, i = 1, \ldots, 2000\}$ as follows:

- Select 2000 points in the squared area $[-2, 2] \times [-2, 2]$ of the $\mathbb{R}^2$ space using a uniform distribution.
- All points that lie on the positive side of the curve $x_2^2 - x_1^2 = 0$ are assigned to the class $+1$, while all the others are assigned to class -1.

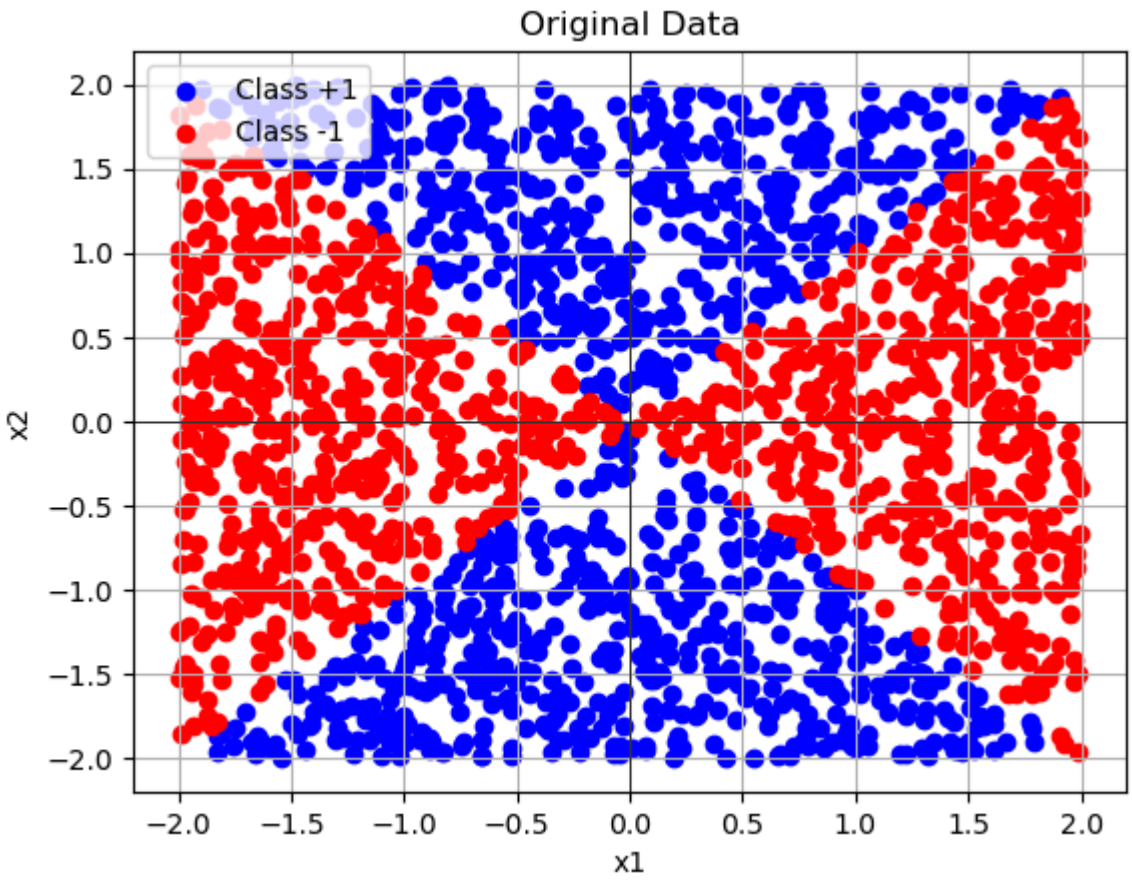
Plot the data using different colors for points from different classes. In the sequel, pretend that you do not know how the data were generated. All you have at your disposal is the data set X.

In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Set random seed for reproducibility
np.random.seed(123)

# Generating 2000 points uniformly in the square [-2, 2] x [-2, 2]
n_points = 2000
x1 = np.random.uniform(-2, 2, n_points)
x2 = np.random.uniform(-2, 2, n_points)

# Classify points based on the curve x2^2 - x1^2 = 0
y = np.where(x2**2 - x1**2 > 0, 1, -1)  # Assign +1 and -1

# Plot the data
plt.figure()
plt.scatter(x1[y == 1], x2[y == 1], color='blue', label='Class +1')
plt.scatter(x1[y == -1], x2[y == -1], color='red', label='Class -1')
plt.title('Original Data')
plt.xlabel('x1')
plt.ylabel('x2')
plt.axhline(0, color='black', lw=0.5)
plt.axvline(0, color='black', lw=0.5)
plt.grid()
plt.legend()
plt.show()
```



(b) Data Transformation

Apply the transformation

$$\phi(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix}$$

on all $\mathbf{x}_i$ of X. Denoting by $\mathbf{x}_i'$ the image of $\mathbf{x}_i$ (that is, $\mathbf{x}_i' = \phi(\mathbf{x}_i)$), we form a new data set

$$X' = \{(y_i, \mathbf{x}_i'), i = 1, \ldots, 2000\}$$

In [8]:
```
# Transformation phi(x) = [x1^2, x2^2]
x1_transformed = x1 ** 2
x2_transformed = x2 ** 2

# Creating new DataFrame
data_transformed = pd.DataFrame({'y': y, 'x1_transformed': x1_transformed, 'x2_trar
data_transformed.head()
```

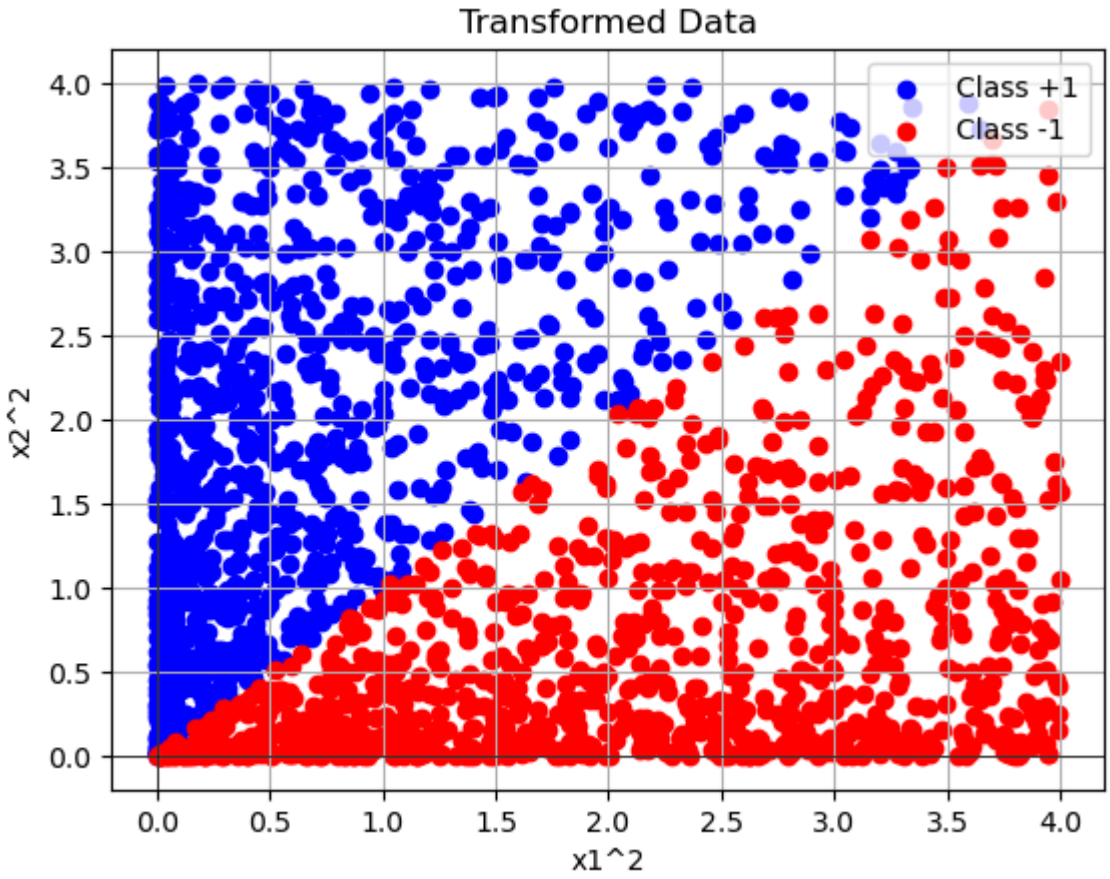
Out[8]:

	y	x1_transformed	x2_transformed
0	-1	0.617602	0.094825
1	-1	0.731782	0.183121
2	1	1.193762	3.434333
3	1	0.042131	2.359537
4	-1	0.770666	0.498324

(c) Plotting Transformed Data

Plot the $\mathbf{x}_i'$ using different colors for points from different classes and compare the resulting plot with that of (a). Comment on the differences between the two plots.

In [9]:
```
# Plotting the transformed data
plt.figure()
plt.scatter(data_transformed[data_transformed['y'] == 1]['x1_transformed'],
            data_transformed[data_transformed['y'] == 1]['x2_transformed'], color='
plt.scatter(data_transformed[data_transformed['y'] == -1]['x1_transformed'],
            data_transformed[data_transformed['y'] == -1]['x2_transformed'], color=
plt.title('Transformed Data')
plt.xlabel('x1^2')
plt.ylabel('x2^2')
plt.axhline(0, color='black', lw=0.5)
plt.axvline(0, color='black', lw=0.5)
plt.grid()
plt.legend()
plt.show()
```



(d) Linear Model Estimation

Adopting the linear model assumption in the transformed space and the sum of error squares criterion, estimate the parameters of the model that minimize this criterion. The model can be expressed mathematically as:

$$\hat{y} = \mathbf{w}^T \mathbf{x}'$$

where $\mathbf{w}$ represents the parameters of the model, and the error squares criterion is given by:

$$E(\mathbf{w}) = \sum_{i=1}^{2000} (y_i - \hat{y}_i)^2$$

In [10]:
```
# Preparing the input features and target variable
X_transformed = np.column_stack((np.ones(n_points), x1_transformed, x2_transformed)
y_transformed = data_transformed['y']

# Performing Ordinary Least Squares (OLS) to estimate theta
theta_hat = np.linalg.inv(X_transformed.T @ X_transformed) @ X_transformed.T @ y_tr
theta_hat
```

Out[10]:
```
array([-0.00197116, -0.45545489,  0.47063574])
```

The new θ values are θ0=-0.001, θ1=-0.455, θ2=0.470

In [ ]: