AI61003 Linear Algebra for AI & ML
Assignment 02 - Problem 09

The code and corresponding outputs are attached on the last page.

(a) The confusion matrix obtained for the 500 random samples on which the model was trained is –

$$
\begin{array}{c} \\ y=+1 \\ y=-1 \\ \\ \end{array}
\begin{array}{cc} \multicolumn{2}{c}{\text{Prediction}} \\ \left[\begin{array}{cc} 241 & 14 \\ 5 & 240 \end{array}\right] \\ \hat{y}=+1 & \hat{y}=-1 \end{array}
\ \text{outcome}
$$

$$\text{error rate} = (N_{fp} + N_{fn}) / N$$
$$= (14+5)/500 = 3.8\%$$

(b) The parameters obtained were θ_p, where

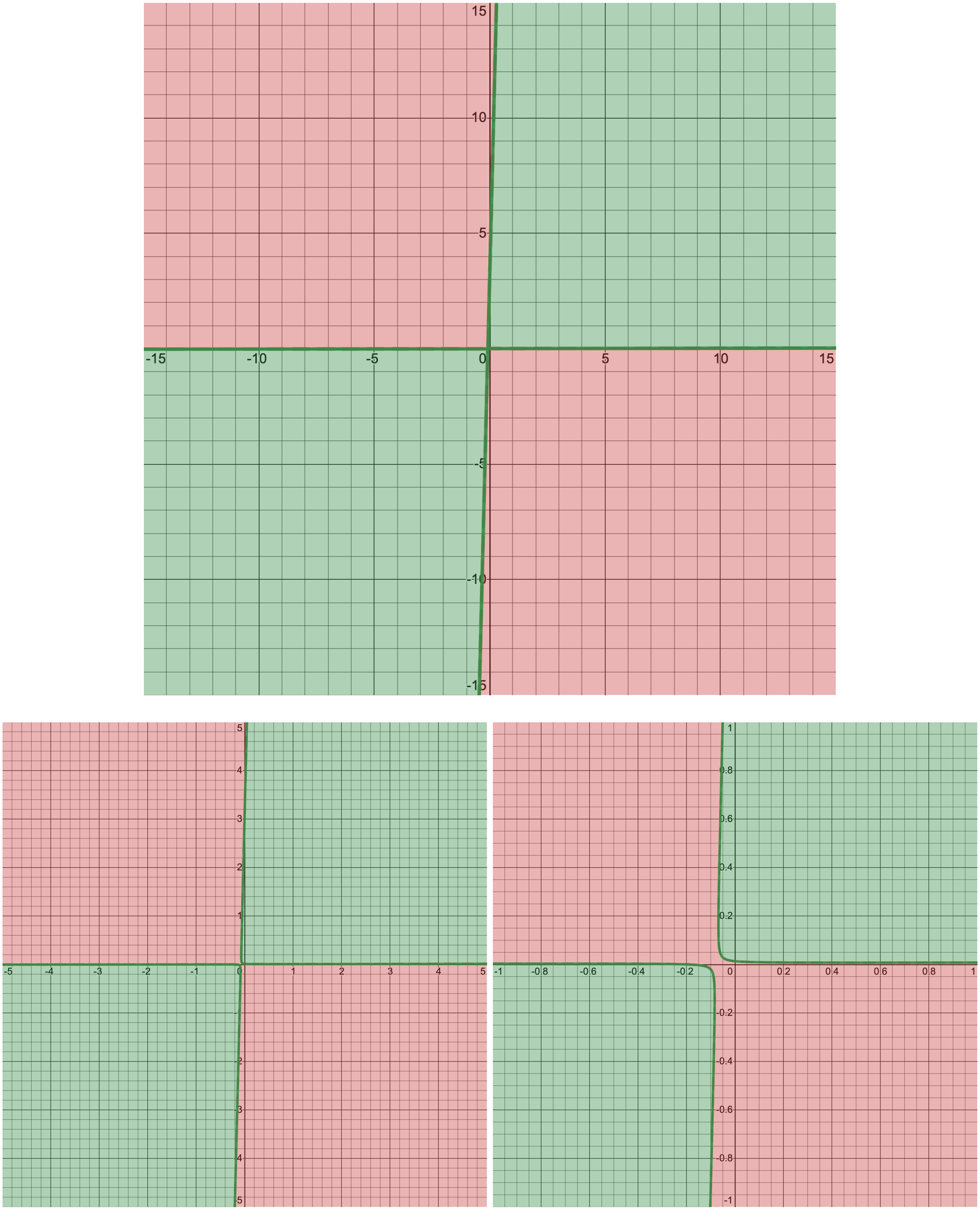
$$\theta_p = [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5 \ \theta_6 \ \theta_7]^T$$

$$\theta_p = \begin{bmatrix} -6.640 \times 10^{-4} \\ -3.745 \times 10^{-3} \\ 4.966 \times 10^{-2} \\ 6.517 \times 10^{-1} \\ -1.049 \times 10^{-3} \\ -1.616 \times 10^{-2} \end{bmatrix}$$

The regions in $\mathbb{R}^2$ plane where $\hat{f}(x) = 1$ and $\hat{f}(x) = -1$ are given by R_+, R_- respectively where,

$$R_+ = \{(x,y) \mid \theta_p^T (1 \ x \ y \ xy \ x^2 \ y^2) \geq 0\}$$
$$R_- = \{(x,y) \mid \theta_p^T (1 \ x \ y \ xy \ x^2 \ y^2) < 0\}$$

The region R_+ is marked in green and R_- is marked in red. The $\mathbf{R}^2$ plane is shown in 3 scales.



The region R_+ is very close to the region consisting of only the first and the third quadrants. Similarly, the region R_- is very close to the region consisting of only the second and the fourth quadrants. This implies that the model $\hat{f}$ precisely fits the actual model $f = xy$.

DATE
PAGE

The regions $R+$ and $R-$ where shown on the $\mathbb{R}^2$ plane on the last page.

(c) Yes, $g = x_1 x_2$ classifies the generated points with zero error.
This is because the prediction $\hat{y} = +1$ iff $\tilde{y} = x_1 x_2 \geq 0$ iff $y = +1$ (by definition of $y^{(i)}$). ·lly applies for $\hat{y} = -1$. (also verified practically in the code).

The parameters of the actual model g are $\theta_a = [\theta_1 \; \theta_2 \; \theta_3 \; \theta_4 \; \theta_5 \; \theta_6 \; \theta_7]^T$

$$\theta_a = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad \theta_p = \begin{bmatrix} -6.640 \\ -37.450 \\ 496.563 \\ 6516.706 \\ -10.486 \\ -161.655 \end{bmatrix} \times 10^{-4}$$

In θ_a, $(\theta_a)_4 = 1$ and rest are 0.
In θ_p, $(\theta_p)_4 >> (\theta_p)_i$ for $i \neq 4$.
So on comparing θ_a and θ_p, we find out that θ_p assings maximum weightage to the $x_1 x_2$ basis function, that is actually true as per the actual model parameters θ_a. The model $\text{sgn}(\tilde{f})$, $\hat{f}$, won't change if θ_p is scaled by $1/(\theta_p)_4$.
So $\theta_p' = \frac{1}{(\theta_p)_4} \theta_p$. ($\because 1/(\theta_p)_4$ is +ve)

$$\theta_p' = \begin{bmatrix} -0.00102 \\ -0.00575 \\ 0.07620 \\ 1.00000 \\ -0.00161 \\ -0.02481 \end{bmatrix}$$

Clearly θ_p' is very close to θ_a.

In [1]:
```python
import numpy as np
```

In [2]:
```python
data = [] # draw 500 random 2d vectors from a standrd normal distribution
for _ in range(500) :
    sample = np.random.normal(0, 1, size = (2))
    if sample[0] * sample[1] >= 0 : data.append((sample, 1))
    else : data.append((sample, -1))
```

In [3]:
```python
# This function computes and returns the LS solution of Ax = b using
# the direct formula, i.e, x_hat = ((A_t.A)_inv).(A_t).b
# where X_t is the transpose and X_inv is the inverse of X
def Least_Squares_Solution ( A , b ) :
    t = np.matmul(np.transpose(A), A)
    t = np.linalg.inv(t)
    t = np.matmul(t, np.transpose(A))
    x_hat = np.matmul(t, b)
    return x_hat

# This function returns the Vandermonde matrix for the given data samples.
# There are 6 basis functions, as given in the problem statement. So
# for 500 samples, the matrix will be of size 500x6.
def Vandermonde_Matrix ( data ) :
    mat = []
    for x, _ in data :
        f = [1, x[0], x[1], x[0]*x[1], x[0]**2, x[1]**2]
        mat.append(f)
    return np.array(mat)

# This function fits a binary classification model and returns the corresponding parameters.
# Each data sample belongs to either +1/"positive" class or 1/"negative" class.
def Polynomial_Classification_Model ( data ) :
    A = Vandermonde_Matrix(data)
    y = [ t for _, t in data ]
    return Least_Squares_Solution(A, y)
```

In [4]:
```python
# This function constructs the confusion matrix for the given data samples with respect
# to the model defined by the given parameters "param".
# If confusion matrix is cf_mat, cf_mat[c][pred] = the no. of data samples belonging to
# the true class c that were classified into the class pred by the trained model.
def Confusion_Matrix ( data , param ) :
    cf = np.zeros((2, 2))
    for x, y in data :
        v = np.array([1, x[0], x[1], x[0]*x[1], x[0]**2, x[1]**2])
        f = np.dot(v, param)
        f_hat = 1
        if f < 0 : f_hat = -1
        cf[int((1-y)/2)][int((1-f_hat)/2)] += 1
    return cf
```

In [5]:
```python
p = Polynomial_Classification_Model(data)
p # the parameters of the fitted-model, i.e, [theta1, theta2, theta3, theta4, theta5, theta6]
```

Out[5]:
```
array([-0.00066402, -0.00374501,  0.0496563 ,  0.65167058, -0.00104865,
       -0.01616548])
```

In [6]:
```python
cf_mat = Confusion_Matrix(data, p)
cf_mat # confusion matrix (needed to compute the error rate)
```

Out[6]:
```
array([[241.,  14.],
       [  5., 240.]])
```

In [8]:
```python
false_pos = cf_mat[1][0]
false_neg = cf_mat[0][1]
error_rate = (false_pos + false_neg) / 500
print(' Error Rate :', round(error_rate, 5))
```

```
 Error Rate : 0.038
```

In [9]:
```python
# Find the confusion matrix corresponding to the model governed
# by the polynomial g = x1.x2
cf_mat_g = np.zeros((2, 2))
for x, y in data :
    f = x[0] * x[1]
    f_hat = 1
    if f < 0 : f_hat = -1
    cf_mat_g[int((1-y)/2)][int((1-f_hat)/2)] += 1
cf_mat_g
```

Out[9]:
```
array([[255.,   0.],
       [  0., 245.]])
```

In [10]:
```python
false_pos_g = cf_mat_g[1][0]
false_neg_g = cf_mat_g[0][1]
error_rate_g = (false_pos_g + false_neg_g) / 500
print(' Error Rate of g :', round(error_rate_g, 5))
# Hence verified that g = x1.x2 polynomial function will classify
# the data with zero error rate.
```

```
 Error Rate of g : 0.0
```