Homework Assignment 2

Dillon Ford: A16092047

Imports

In [1]:
```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
```

Images

In [2]:
```
Figure_1 = Image.open('/Users/Dillon/Desktop/Winter_2020/COGS_118A/Assignment2/figure_1_decision_boundary_1.png')
Figure_2 = Image.open('/Users/Dillon/Desktop/Winter_2020/COGS_118A/Assignment2/figure_2_decision_boundary_2.png')
Figure_3 = Image.open('/Users/Dillon/Desktop/Winter_2020/COGS_118A/Assignment2/figure_3_decision_boundary_3.png')
Figure_4 = Image.open('/Users/Dillon/Desktop/Winter_2020/COGS_118A/Assignment2/figure_4_decision_boundary_4.png')
Figure_2_1 = Image.open('/Users/Dillon/Desktop/Winter_2020/COGS_118A/Assignment2/figure_2_1_my_decision_boundary.png')
Figure_2_4 = Image.open('/Users/Dillon/Desktop/Winter_2020/COGS_118A/Assignment2/figure_2_4_decision_boundary.png')
```

1. (10 points) Conceptual Questions

(1.1) Is the following statement true or false?

$f(x)$ is linear with respect to x, given $f(x) = w_0 + w_1 x + w_2 x^2$ where $x, w_0, w_1, w_2 \in R$.

$\boxed{False}$, a linear function, $f(x)$, is a polynomial function in which the variable x has a degree of at most one.

(1.2) “One-hot encoding” is a standard technique that turns categorical features into general real numbers. If we have a dataset S containing m data points where each data point has 1 categorical feature. Specifically, this categorical feature has k possible categories. Thus, the shape of the one-hot encoding matrix that represents the dataset S is:

A. kxk
B. $1xk$
C. $\boxed{mxk} \leftarrow$
D. mxm

Answer: C

(1.3) Assume we have a binary classification model:

$$f(x) = \begin{cases} +1, w \cdot x + b \geq 0 \\ -1, w \cdot x + b \leq 0 \end{cases}$$

where the feature vector $x = (x_1, x_2) \in R^2$, bias $b \in R$, weight vector $w = (w_1, w_2) \in R^2$. The decision boundary of the classification model is:

$w \cdot x + b = 0$

(a) If the predictions of the classifier f and its decision boundary $w \cdot x + b = 0$ are shown in **Figure 1**, which one below can be a possible solution of weight vector w and bias b?

In [3]:
```
Figure_1
```

Out[3]:

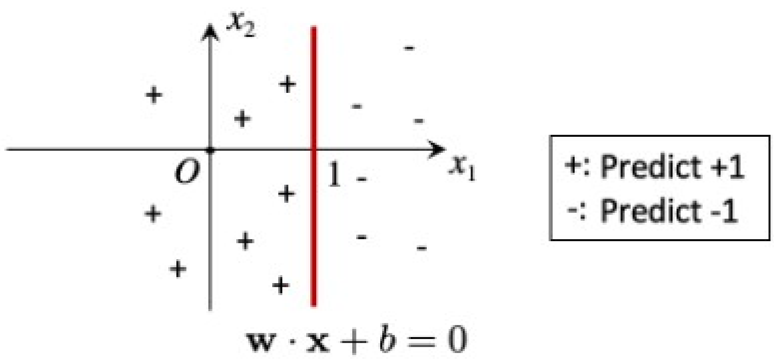


Figure 1: Decision Boundary 1

A. $w = (+1, 0), b = -1$
B. $\boxed{w = (-1, 0), b = +1} \leftarrow$
C. $w = (+1, 0), b = +1$
D. $w = (0, -1), b = -1$

Answer: B

(b) If the predictions of the classifier f and its decision boundary $w \cdot x + b = 0$ are shown in **Figure 2**, which one below can be a possible solution of weight vector w and bias b?

In [4]:
```
Figure_2
```

Out[4]:

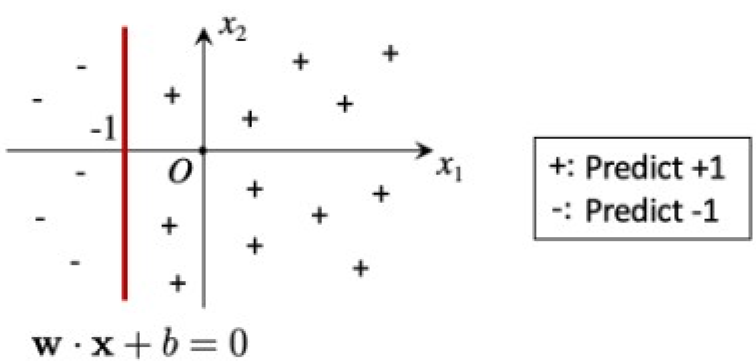


Figure 2: Decision Boundary 2

A. $w = (+1, 0), b = -1$
B. $w = (-1, 0), b = +1$
C. $\boxed{w = (+1, 0), b = +1} \leftarrow$
D. $w = (0, -1), b = -1$

Answer: C

(1.4) Choose the most significant difference between regression and classification:

A. unsupervised learning vs. supervised learning.
B. $\boxed{\text{prediction of continuous values vs. prediction of class labels.}} \leftarrow$
C. features are not one-hot encoded vs features are one-hot encoded.
D. none of the above.

Answer: B

2. (25 points) Decision Boundary

2.1 (3 points)

We are given a classifier that performs classification in R^2 (the space of data points with 2 features (x_1, x_2)) with the following decision rule:

$$h(x_1, x_2) = \begin{cases} 1, \text{if } 2x_1 + 4x_2 - 8 \geq 0 \\ 0, \text{otherwise} \end{cases}$$

1. Draw the decision boundary of the classifier and shade the region where the classifier predicts
Make sure you have marked the x_1 and x_2 axes and the intercepts on those axes.

$2x_1 + 4x_2 - 8 \geq 0$

when $x_1 = 0$,

$4x_2 - 8 = 0$
$\Rightarrow x_2 = 2$

When $x_2 = 0$,

$2x_1 - 8 = 0$
$\Rightarrow x_1 = 4$

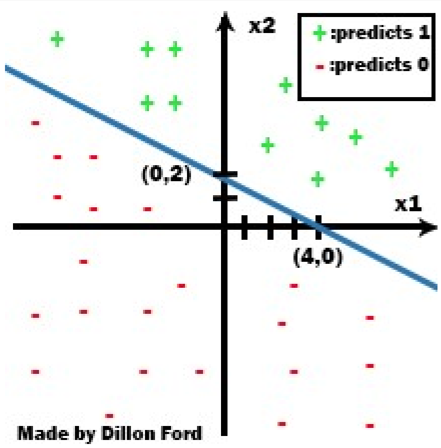
This gives us the intercepts, $(x_1, x_2) = (0, 2), (4, 0)$

As $2x_1 + 4x_2 - 8 \geq 0$ has the prediction 1
otherwise 0

We have the graph

In [5]:
```
Figure_2_1
```

Out[5]:



2.2 (9 points)

We are given a classifier that performs classification in R^2 (the space of data points with 2 features (x_1, x_2)) with the following decision rule:

$$h(x_1, x_2) = \begin{cases} 1, \text{if } w_1x_1 + w_2x_2 + b \geq 0 \\ 0, \text{otherwise} \end{cases}$$

Here, the normal vector w of the decision boundary is normalized, i.e.:

$$\|\mathbf{w}\|_2 = \sqrt{w_1^2 + w_2^2} = 1$$

1. Compute the parameters w_1, w_2 and b for the decision boundary in **Figure 3**. Please make sure the predictions from the obtained classifier are consistent with **Figure 3**.

Hint: Please use the intercepts in the **Figure 3** to find the relation between w_1, w_2 and b. Then, substitute it into the normalization constraint to solve for parameters.

In [6]:
```
Figure_3
```

Out[6]:

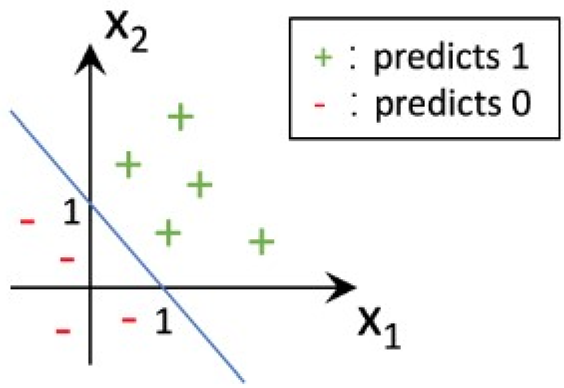


Figure 3: Decision boundary to solve for parameters.

From the graph we can see two intercepts, one on the x_1 axis and one on the x_2 axis

Where $x_1 = 1$ and $x_2 = 0$ (point on x_1 axis) i.e. the point $(1, 0)$
Where $x_1 = 0$ and $x_2 = 1$ (point on x_1 axis) i.e. the point $(0, 1)$

For, $w_1x_1 + w_2x_2 + b = 0$
Substituting, $(1, 0) \Rightarrow w_1 + b = 0$ .....(1)
Substituting, $(0, 1) \Rightarrow w_2 + b = 0$ .....(2)

Subtracting equations (1) and (2) yields,
$w_1 = w_2$

Since, $\sqrt{w_1^2 + w_2^2} = 1$, and $w_1 = w_2$ then it becomes,

$\sqrt{2w_1^2} = 1$

$\Rightarrow 2w_1^2 = 1$
$\Rightarrow w_1 = w_2 = \frac{1}{\sqrt{2}}$

Solving for b from equation (1) or (2) yields,

$\frac{1}{\sqrt{2}} + b = 0$
$\Rightarrow b = -\frac{1}{\sqrt{2}}$

Hence our parameters are,

$$\boxed{w_1 = w_2 = \frac{1}{\sqrt{2}} \text{ and } b = -\frac{1}{\sqrt{2}}} \leftarrow$$

2. Use parameters from the above question to compute predictions for the following two data points:
$A = (3, 6), B = (1, -4)$.

For the data point $A = (3, 6)$ we have,

$\frac{3}{\sqrt{2}} + \frac{6}{\sqrt{2}} - \frac{1}{\sqrt{2}} \geq 0$

$\Rightarrow 4\sqrt{2} \geq 0$

$\boxed{\text{The predicted label for the data point } A = 1} \leftarrow$

For the data point $B = (1, -4)$ we have,

$\frac{1}{\sqrt{2}} - \frac{4}{\sqrt{2}} - \frac{1}{\sqrt{2}} \geq 0$

$\Rightarrow -2\sqrt{2} \ngeq 0$

$\boxed{\text{The predicted label for the data point } B = 0} \leftarrow$

2.3 (10 points)

We are given a classifier that performs classification in R^3 (the space of data points with 3 features (x_1, x_2, x_3)) with the following decision rule:

$$h(x_1, x_2, x_3) = \begin{cases} 1, \text{if } w_1x_1 + w_2x_2 + w_3x_3 + b \geq 0 \\ 0, \text{otherwise} \end{cases}$$

Here, the normal vector w of the decision boundary is normalized, i.e.:

$$\|\mathbf{w}\|_2 = \sqrt{w_1^2 + w_2^2 + w_3^2} = 1$$

In addition, we set $b \leq 0$ to have a unique equation for the decision boundary.

1. Compute the parameters w_1, w_2, w_3 and b for the decision boundary that passes through three points $A = (3, 2, 4), B = (-1, 0, 2), C = (4, 1, 5)$ in **Figure 4.**

Hint: Please use the intercepts in the **Figure 4** to find the relation between $w1$, $w2$, $w3$ and b. Then, substitute it into the normalization constraint to solve for parameters.

In [7]:
```
Figure_4
```

Out[7]:

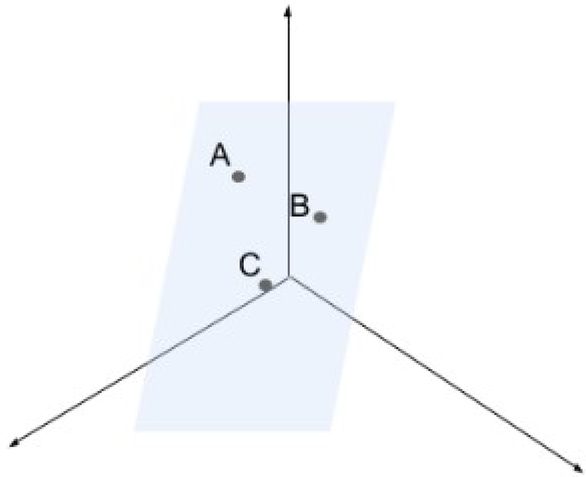


Figure 4: Decision boundary to solve the parameters.

Equation of a plane passing through,

$A = (3, 2, 4)$
$B = (-1, 0, 2)$
$C = (4, 1, 5)$

$B - A = (-4, -2, -2) = \overrightarrow{v_1}$
$C - B = (5, 1, 3) = \overrightarrow{v_2}$

$\overrightarrow{v_1} \times \overrightarrow{v_2} = (-2 \cdot 3 - (-2 \cdot 1) - 2 \cdot 5 - 4 \cdot 1 - (-2 \cdot 5)) = (-4, 2, 6)$

$\Rightarrow (w_1, w_2, w_3) = \frac{(-4,2,6)}{\sqrt{4^2+2^2+6^2}} = \frac{-4}{\sqrt{56}}, \frac{2}{\sqrt{56}}, \frac{6}{\sqrt{56}}$

$\sqrt{\frac{-4}{\sqrt{56}}^2 + \frac{2}{\sqrt{56}}^2 + \frac{6}{\sqrt{56}}^2} = 1$

$\Rightarrow \sqrt{\frac{2}{7} + \frac{1}{14} + \frac{9}{14}} = 1$

$\Rightarrow \sqrt{1} = 1 \Rightarrow 1 = 1$

$$\boxed{\Rightarrow w_1 = \frac{-4}{\sqrt{56}}, w_2 = \frac{2}{\sqrt{56}}, w_3 = \frac{6}{\sqrt{56}}} \leftarrow$$

as, $B = (-1, 0, 2) \Rightarrow (x_1, x_2, x_3)$

$\Rightarrow w_1 x_1 + w_2 x_2 + w_3 x_3 = -b$

$\Rightarrow \frac{4}{\sqrt{56}} + 0 + \frac{12}{\sqrt{56}} = -b$

$$\boxed{\Rightarrow b = \frac{-16}{\sqrt{56}}} \leftarrow$$

2. Use parameters from the above question to compute predictions for the following two data points: $D = (0, 0, 0), E = (1, 0, 5)$.

For the data point $D = (0, 0, 0)$ we have,

$0 + 0 + 0 - \frac{-16}{\sqrt{56}}$

$\Rightarrow -\frac{-16}{\sqrt{56}} \ngeq 0$

$\boxed{\text{The predicted label for the data point } D = 0} \leftarrow$

For the data point $E = (1, 0, 5)$ we have,

$\frac{-4}{\sqrt{56}} + 0 + \frac{30}{\sqrt{56}} - \frac{16}{\sqrt{56}}$

$\Rightarrow \frac{10}{\sqrt{56}} \geq 0$

$\boxed{\text{The predicted label for the data point } E = 1} \leftarrow$

2.4 (3 points)

We are given a classifier that performs classification in R^2 (the space of data points with 2 features (x_1, x_2)) with the following decision rule:

$$h(x_1, x_2) = \begin{cases} 1, \text{if } x_1^2 + x_2^2 - 10 \geq 0 \\ 0, \text{otherwise} \end{cases}$$

1. Draw the decision boundary of the classifier and shade the region where the classifier predicts
Make sure you have marked the x_1 and x_2 axes and the intercepts on those axes.

$x_1^2 + x_2^2 - 10 \geq 0$

when $x_1 = 0$,

$\Rightarrow x_2 = \sqrt{10}$

When $x_2 = 0$,

$\Rightarrow x_1 = \sqrt{10}$

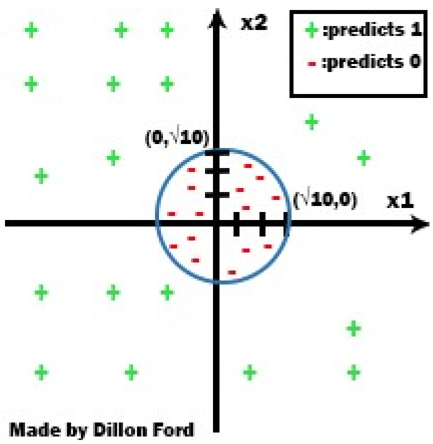
This gives us the intercepts, $(x_1, x_2) = (0, \sqrt{10}), (\sqrt{10}, 0)$

As $x_1^2 + x_2^2 - 10 \geq 0$ has the prediction 1
otherwise 0

We have the graph

In [8]:
```
Figure_2_4
```

Out[8]:



3. (10 points) Derivatives

3.1 Function Defined by Scalars

1. (3 points) Given a function $f(w) = (y_1 + wx_1)^2$ where $(x_1, y_1) = (3, 4)$ represents a data point, derive $\frac{\partial f(w)}{\partial w}$

$\frac{\partial f(w)}{\partial w} = 2(y_1 + wx_1)(x_1)$

such that, $(x_1, y_1) = (3, 4)$ we have,

$\frac{\partial f(w)}{\partial w} = 2(4 + 3w)(3)$

$$\boxed{\frac{\partial f(w)}{\partial w} = 6(4 + 3w)} \leftarrow$$

2. (3 points) Given a function $f(w) = \sum_{i \epsilon [1,2]} (y_i - wx_i)^2$ where $(x_1, y_1) = (1, 1), (x_2, y_2) = (2, 3)$ are two data points, derive $\frac{\partial f(w)}{\partial w}$

$f(w) = (y_1 - wx_1)^2 + (y_2 - wx_2)^2$

$\frac{\partial f(w)}{\partial w} = 2(y_1 - wx_1)(-x_1) + 2(y_2 - wx_2)(-x_2)$

For $(x_1, y_1) = (1, 1)$ and $(x_2, y_2) = (2, 3)$ we have,

$\frac{\partial f(w)}{\partial w} = 2(1 - w)(-1) + 2(3 - 2w)(-2)$

$$\boxed{\frac{\partial f(w)}{\partial w} = 2(w - 1) + 4(2w - 3)} \leftarrow$$

3.2 Function Defined by Vectors

1. (4 points) Given a function $f(w) = (y - wx)^T (y - wx)$ where $x = [1, 2]^T$ and $y = [1, 3]^T$, derive $\frac{\partial f(w)}{\partial w}$.
Note: In $f(w), w \epsilon R$ is still a scalar.

$$f(w) = \left(\begin{bmatrix}1\\3\end{bmatrix} - w\begin{bmatrix}1\\2\end{bmatrix}\right)^T \left(\begin{bmatrix}1\\3\end{bmatrix} - w\begin{bmatrix}1\\2\end{bmatrix}\right)$$

$$f(w) = \left(\begin{bmatrix}1\\3\end{bmatrix} - \begin{bmatrix}w\\2w\end{bmatrix}\right)^T \left(\begin{bmatrix}1\\3\end{bmatrix} - \begin{bmatrix}w\\2w\end{bmatrix}\right)$$

$$f(w) = \begin{bmatrix}1-w\\3-2w\end{bmatrix}^T \begin{bmatrix}1-w\\3-2w\end{bmatrix}$$

$$f(w) = [1-w,\ 3-2w] \begin{bmatrix}1-w\\3-2w\end{bmatrix}$$

$$f(w) = (1-w)(1-w) + (3-2w)(3-2w)$$

$$f(w) = (1-w)^2 + (3-2w)^2$$

$$f(w) = (1-2w+w^2) + (9-12w+4w^2)$$

$$f(w) = 10 - 14w - 5w^2$$

$$\frac{\partial f(w)}{\partial w} = 0 - 14 + 10w$$

$$\boxed{\frac{\partial f(w)}{\partial w} = 10w - 14} \leftarrow$$

4. (9 points) Concepts

Select the correct option(s). Note that there might be multiple correct options

1. For two monotonically increasing functions f(x) and g(x):

A. $\boxed{f(x) + g(x) \text{ is always monotonically increasing.}} \leftarrow$

B. $f(x) - g(x)$ is always monotonically increasing.

C. $f(x^2)$ is always monotonically increasing.

D. $\boxed{f(x^3) \text{ is always monotonically increasing.}} \leftarrow$

Answer: A, D

2. For a function $f(x) = x(10 - x)$, $x \epsilon R$, please choose the correct statement(s) below:

$f'(x) = 10 - 2x = 0$

$\Rightarrow x = 5, > 0 \Rightarrow$ maxima at $x = 5$

$f''(x) = -2, < 0 \Rightarrow$ no minima

$\text{argmax}_x f(x) = 5$

$\max_x f(x) = f(5) = 5 \cdot (5) = 25$

A. $\boxed{\text{argmax}_x f(x) = 5.} \leftarrow$
B. $\text{argmin}_x f(x) = 25.$
C. $\min_x f(x) = 5.$
D. $\boxed{\max_x f(x) = 25.} \leftarrow$

Answer: A,D

3. Assume we have a function $f(x)$ which is differentiable at every $x \epsilon R$. There are three properties that describe the function $f(x)$:

(1) $f(x)$ is a convex function.
(2) When $x = x_0, f'(x_0) = 0.$
(3)$f(x_0)$ is a global minimum of $f(x)$.

Which one of the following statements is wrong?
Hint: You can use a failure case to disprove a statement.

A. Given (1) and (2), we can prove that (3) holds.
B. $\boxed{\text{Given (2) and (3), we can prove that (1) holds.}} \leftarrow$
C. Given (1) and (3), we can prove that (2) holds.

Answer: B

5 (4 points) Argmin and Argmax

An unknown estimator is given an estimation problem to find the minimizer and maximizer of the objective function $G(w) \epsilon (0, 2]$:

$(w_a, w_b) =$ (argmin$_w G(w)$, argmax$_w G(w)$)......(1)

The solution to Eq.1 by the estimator is $(w_a, w_b) = (10, 20)$.

Given this information, please obtain the value of w^* such that:

w^*= argmin$_w [10 - 4 \times ln(G(w))]$......(2)

$ln(x)$ is a monotonic increasing function with $ln(x) > 0$ for $x > 0$
$ln(G(w))$ is a monotonic increasing function
$-ln(x)$ is a monotonic decreasing function
$-4ln(x)$ is a monotonic decreasing function

so, $10 - 4 \times ln(x)$ is minimum when G(w) is maximum.

$(w_a, w_b) = (10, 20)$ and so,

$\boxed{w^* = w_b = 20} \leftarrow$

6. (12 points) Data Manipulation

In this question, we still use the Iris dataset from Homework 1. In fact, you can see the shape of array X is (150,4) by running **X.shape**, which means it contains 150 data points whereeach has 4 features. Here, we will perform some basic data manipulation and calculate some statistics:

1. Divide array X evenly to five subsets of data points:
Group 1: 1st to 30th data point,
Group 2: 31st to 60th data point,
Group 3: 61st to 90th data point,
Group 4: 91st to 120th data point,
Group 5: 121st to 150th data point.

Then calculate the mean of feature vectors in each group. Your results should be five 4-dimensional vectors (i.e. shape of NumPy array can be $(4, 1), (1, 4)$ or $(4,))$

In [9]:
```
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

In [10]:
```
# divide the dataset into five subsets
group_1 = X[0:30,:]
group_2 = X[30:60,:]
group_3 = X[60:90,:]
group_4 = X[90:120,:]
group_5 = X[120:150,:]
print('Shape of group subsets:',group_1.shape, group_2.shape, group_3.s
hape, group_4.shape, group_5.shape)
```

```
Shape of group subsets: (30, 4) (30, 4) (30, 4) (30, 4) (30, 4)
```

In [11]:
```
# calculate the mean of feature vectors in each group
group_1_mean = np.mean(group_1, axis=0)
group_2_mean = np.mean(group_2, axis=0)
group_3_mean = np.mean(group_3, axis=0)
group_4_mean = np.mean(group_4, axis=0)
group_5_mean = np.mean(group_5, axis=0)
print('Mean of Group 1 feature vectors:',group_1_mean )
print('Mean of Group 2 feature vectors:',group_2_mean )
print('Mean of Group 3 feature vectors:',group_3_mean )
print('Mean of Group 4 feature vectors:',group_4_mean )
print('Mean of Group 5 feature vectors:',group_5_mean )
```

```
Mean of Group 1 feature vectors: [5.02666667 3.45       1.47333333 0.
24666667]
Mean of Group 2 feature vectors: [5.35       3.22       2.42       0.
62333333]
Mean of Group 3 feature vectors: [5.98 2.75 4.3  1.34]
Mean of Group 4 feature vectors: [6.25333333 2.85666667 5.11333333 1.
77333333]
Mean of Group 5 feature vectors: [6.60666667 3.01       5.48333333 2.
01333333]
```

2. Remove 2nd and 3rd features from array X, resulting a 150×2 matrix. Then calculate the mean of all feature vectors. Your result should be a 2-dimensional vector.

In [12]:
```
X_1 = X[:,[0,3]]
print('shape of X:',X_1.shape)
print('mean of feature vectors:',np.mean(X_1, axis=0))
```

```
shape of X: (150, 2)
mean of feature vectors: [5.84333333 1.19933333]
```

3. Remove last 10 data points from array X, resulting a 140×4 matrix. Then calculate the mean of feature vectors. Your result should be a 4-dimensional vector.

In [13]:
```
X_2 = X[:-10,:]
print('shape of X:',X_2.shape)
print('mean of feature vectors:',np.mean(X_2, axis=0))
```

```
shape of X: (140, 4)
mean of feature vectors: [5.8        3.05928571 3.64571429 1.13
]
```

7. (15 points) Training vs. Testing Errors

In this problem, we are given two trained predictive models on a modified Iris dataset. Each data point (x, y) has a feature vector $x_i \, \epsilon \, R^4$ and its corresponding label $y_i \, \epsilon \, [0, 1]$, where $i \, \epsilon \, [1, 2, \ldots, 150]$. To predict on the new data, here we consider two types of model: a regression model and a classification model. The regression model is trained to predict a real number, while the classification model applies a threshold to the output of the regression model, converting the real number into a binary value.

The regression model is as followed:

$$\hat{y}_i(x_i) = w^T x_i + b$$

The classifier is as followed:

$$h(x_i) = \begin{cases} 1, \text{if } \hat{y}_i(x_i) \geq \frac{1}{2} \\ 0, \text{otherwise} \end{cases}$$

where $w = [0.1297, 0.1225, -0.1171, 0.6710]^T, b = -1.1699.$

The regression error is defined as:

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

and the classification error is defined as:

$$\frac{1}{n}\sum_{i=1}^{n} 1(h(x_i) \neq y_i)$$

where n is the number of data points.

The data as well as the split of training and testing set are given in the Jupyter notebook we provided. **You should not use the scikit-learn library.**

Please download the notebook training_test_errors.ipynb from the course website and fill in the missing blanks. Follow the instructions in the skeleton code and report:

•Training error of the regression model.
•Testing error of the regression model.
•Training error of the classification model.
•Testing error of the classification model.

Load the Iris dataset

In [14]:

```
# Iris dataset.
iris = datasets.load_iris()      # Load Iris dataset.

X = iris.data                    # The shape of X is (150, 4), which mea
ns
                                 # there are 150 data points, each data
point
                                 # has 4 features.

# Here for convenience, we divide the 3 kinds of flowers into 2 groups:
#      Y = 0 (or False):  Setosa (original value 0) / Versicolor (origin
al value 1)
#      Y = 1 (or True):   Virginica (original value 2)

# Thus we use (iris.target > 1.5) to divide the targets into 2 groups.
# This line of code will assign:
#    Y[i] = True  (which is equivalent to 1) if iris.target[k]  > 1.5
(Virginica)
#    Y[i] = False (which is equivalent to 0) if iris.target[k] <= 1.5
(Setosa / Versicolor)

Y = (iris.target > 1.5).reshape(-1,1) # The shape of Y is (150, 1), whi
ch means
                                 # there are 150 data points, each data
point
                                 # has 1 target value.

X_and_Y = np.hstack((X, Y))      # Stack them together for shuffling.
np.random.seed(1)                # Set the random seed.
np.random.shuffle(X_and_Y)       # Shuffle the data points in X_and_Y ar
ray

print(X.shape)
print(Y.shape)
print(X_and_Y[0])                # The result should be always: [ 5.8
4.   1.2  0.2  0. ]
```

```
(150, 4)
(150, 1)
[5.8 4.  1.2 0.2 0. ]
```

In [15]:

```
# Divide the data points into training set and test set.
X_shuffled = X_and_Y[:,:4]
Y_shuffled = X_and_Y[:,4]

X_train = X_shuffled[:100] # Shape: (100,4)
Y_train = Y_shuffled[:100] # Shape: (100,)
X_test = X_shuffled[100:]  # Shape: (50,4)
Y_test = Y_shuffled[100:]  # Shape: (50,)
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(100, 4)
(100,)
(50, 4)
(50,)
```

In [16]:

```
from sklearn.linear_model import LinearRegression

# let's train a LR model...
# note that this time we let sklearn fit the intercept
# ask yourself why. Ask yourself what could we have done
# to X_and_Y so that we could have used
# = LinearRegression(fit_intercept=False).fit(X_train, Y_train)
# instead of the below line, and got identical results?

pre_defined_weights = LinearRegression().fit(X_train, Y_train)
w = pre_defined_weights.coef_
b = pre_defined_weights.intercept_
```

In [20]:
```
def regression_error(x, y, w, b):

    regression_error = 0
    for i in range(len(x)):

        # TODO: ******** To be filled ********

        # prediction based on x

        y_hat = np.dot(w.transpose(), x[i]) + b

        # regression error, doing the sum
        regression_error += np.square(y_hat-y[i])

    # calculate the mean and square root
    regression_error = np.sqrt(regression_error*(1/len(x)))

    return regression_error

def classification_error(x, y, w, b):
    classification_error = 0

    for i in range(len(x)):

        # TODO: ******** To be filled ********

        # prediction based on x
        h_x = 0

        y_hat = np.dot(w.transpose(), x[i]) + b

        # classification error
        if y_hat >= (1/2):
            h_x = 1
        error = 1*int(h_x != y[i])

        classification_error += error

    # calculate the mean of error
    classification_error = (classification_error)/(len(x))

    return classification_error

print('Training regression errors are:')
print(regression_error(X_train, Y_train, w, b))
print('Testing regression errors are:')
print(regression_error(X_test, Y_test, w, b))

print('Training classification errors are:')
print(classification_error(X_train, Y_train, w, b))
print('Testing classification errors are:')
print(classification_error(X_test, Y_test, w, b))
```

```
Training regression errors are:
0.2792069270624264
Testing regression errors are:
0.33046623492235216
Training classification errors are:
0.06
Testing classification errors are:
0.14
```

8. (15 points) Linear Regression

Assume we are given a dataset $S = \{(x_i, y_i), i = 1, \ldots, n\}$. Here, $x_i \ \epsilon \ R$ is a feature scalar (a.k.a. value of input variable) and $y_i \ \epsilon \ R$ is its corresponding value (a.k.a. value of dependent variable). In this section, we aim to fit data points with a line:

$$y = w_0 + w_1 x \qquad\qquad (3)$$

where $w_0, w_1 \ \epsilon \ R$ are two parameters to determine the line. Next, we measure the quality of fitting by evaluating a sum-of-squares error functiong (w_0, w_1):

$$g(w_0, w_1) = \sum_{i=1}^{n} (w_0 + w_1 x_i - y_i)^2 \qquad (4)$$

When $g(w_0, w_1)$ is near zero, it means the proposed line can fit the dataset and model an accurate relation between x_i and y_i. The best line with parameters (w_0^*, w_1^*) can reach the minimum value of the error function $g(w_0, w_1)$:

$$(w_0^*, w_1^*) = \text{argmin } g(w_0, w_1) \qquad (5)$$

To obtain the parameters of the best line, we will take the gradient of function $g(w_0, w_1)$ and set it to zero. That is:

$$\nabla g(w_0, w_1) = 0 \qquad (6)$$

The solution (w_0^*, w_1^*) of the above equation will determine the best line $y = w_0^* + w_1^* x$ that fits the dataset S.

In reality, we typically tackle this task in a matrix form: First, we represent data points as matrices $X = [x_1, x_2, \ldots, x_n]^T$ and $Y = [y_1, y_2, \ldots, y_n]^T$, where $x_i = [1, x_i]^T$ is a feature vector corresponding to x_i. The parameters of the line are also represented as a matrix $W = [w_0, w_1]^T$. Thus, the sum-of-squares error function $g(W)$ can be defined as (a.k.a. squared L_2 norm):

$$g(W) = \sum_{i=1}^{n} (x_i^T W - y_i)^2 \qquad (7)$$

$$= ||XW - Y||_2^2 \qquad (8)$$

$$= (XW - Y)^T (XW - Y) \qquad (9)$$

Similarly, the parameters $W^* = [w_0^*, w_1^*]^T$ of the best line can be obtained by solving the equation below:

$$\nabla g(W) = \frac{\partial g(W)}{\partial W} = 0 \qquad (10)$$

(a) According to Eq. 8 and 9, compute the gradient of $g(W)$ with respect to W. Your result should be in the form of X, Y and W.

$g(W) = (XW - Y)^T(XW - Y)$
$g(W) = (X^TW^T - Y^T)(XW - Y)$
$g(W) = X^TW^TXW - W^TX^TY - Y^TXW + Y^TY$

Since, W^TX^TY and Y^TXW are 1x1 matrices,

$(W^TX^TY)^T = Y^TXW$
$\Rightarrow W^TX^TY = Y^TXW$

Thus, $g(W) = Y^TY - 2W^TX^TY + W^TX^TXW$

$\nabla g(W) = \nabla(Y^TY - 2W^TX^TY + W^TX^TXW)$
$\nabla g(W) = \nabla(Y^TY) - 2\nabla(W^TX^TY) + \nabla(W^TX^TXW)$

$\nabla(Y^TY) \Rightarrow \frac{\partial}{\partial W}(Y^TY) = 0$
$\nabla(W^TX^TY) \Rightarrow \frac{\partial}{\partial W}(W^TX^TY) = X^TY$
$\nabla(W^TX^TXW) \Rightarrow \frac{\partial}{\partial W}(W^TX^TXW) = 2X^TXW$

$\nabla g(W) = 0 - 2X^TY + 2X^TXW$
$\boxed{\nabla g(W) = 2(X^TXW - X^TY)} \leftarrow$

(b) By setting the answer of part **(a)** to 0, prove the following:

$$W^* = \text{argmin}_w g(W) = (X^TX)^{-1}X^TY \qquad (11)$$

Note: The above formula demonstrates a closed form solution of Eq. 10

$\nabla g(W) = 0$
$\Rightarrow 2(X^TXW - X^TY) = 0$
$\Rightarrow X^TXW - X^TY = 0$
$\Rightarrow X^TXW = X^TY$
$\Rightarrow (X^TX)^{-1}X^TXW = (X^TX)^{-1}X^TY$
$\Rightarrow W = (X^TX)^{-1}X^TY$

So, the parameter of our best line is given by,

$\boxed{W^* = (X^TX)^{-1}X^TY} \leftarrow$

Previously, we define a sum-of-squares error function $g(w_0, w_1) = \sum_{i=1}^{n}(y_i - w_0 - w_1 x_1)^2$ and represent it in a matrix form $g(W) = ||XW - Y||_2^2$. Actually, we can have multiple choices of the error function: For example, we can define a sum-of-absolute error function $h(w_0, w_1)$:

$$h(w_0, w_1) = \sum_{i=1}^{n} |w_0 + w_1 x_i - y_i| \qquad (12)$$

and represent it in a matrix form $h(W)$ (a.k.a. L_1 norm):

$$h(W) = \sum_{i=1}^{n} |x_i^T W - y_i| \qquad (13)$$

$$= ||XW - Y||_1 \qquad (14)$$

(c) According to the Eq. 13, compute the gradient of the error function $h(W)$ with respect to W. Your result should be in the form of x_i, y_i and W.

Hint: Given a function $f(x) \, \epsilon R$, we have:

$$\frac{\partial \mid f(x)|}{\partial W} = \text{sign}(f(x))\frac{\partial f(x)}{\partial x}$$

where,

$$\text{sign}(x) = \begin{cases} 1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}$$

$h(W) = \sum_{i=1}^{n} |x_i^T W - y_i|$

$\nabla h(W) = \frac{\partial |h(W)|}{\partial W} = \sum_{i=1}^{n} sign(x_i^T W - y_i)\frac{\partial h(W)}{\partial W}$

$$\boxed{\nabla h(W) = \sum_{i=1}^{n} (sign(x_i^T W - y_i))x_i} \leftarrow$$