COGS118 - HW2

By Gurpreet Dhillon, PID: A14220554

1) Decision Boundary

1.1 We are given a classifier that performs classification in $\mathbb{R}^2$ (the space of data points with 2 features (x1,x2)) with the following decision rule:

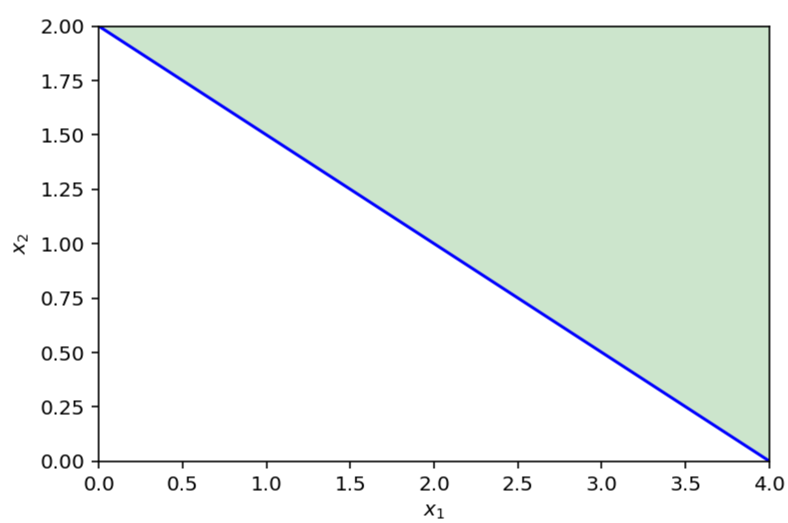
$$h(x1, x2) = \begin{cases} 1, if\ x_1 + 2x_2 - 4 \geq 0 \\ 0,\ Otherwise \end{cases}$$

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

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [42]:
```
x_1 = np.arange(0, 10, 1)
x_2 = 2 -(x_1)/2
plt.plot(x_1,x_2, c='b')
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.xlim([0,4])
plt.ylim([0,2])
plt.fill_between(x_1, x_2, 2, color= 'g', alpha=0.2)
```

Out[42]:
```
<matplotlib.collections.PolyCollection at 0x1a1c74d6d8>
```



1.2 We are given a classifier that performs classification in $\mathbb{R}^2$ (the space of data points with 2 features (x1,x2)) with the following decision rule:

$$h(x1, x2) = \begin{cases} 1, if\ w_1x_1 + w_2x_2 + b \geq 0 \\ 0,\ Otherwise \end{cases}$$

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

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

1. Compute the parameters w_1, w_2 and b for the decision boundary in Figure 1. Please make sure the prediction of the decision boundary you got is consistent with Figure 1.

Hint: Utilize the intercepts in the figure to find the relation between w_1, w_2 and b. Then, substitute it into the normalization constraint to solve the values for the parameters.

$$\sqrt{\left(\frac{1}{\sqrt{2}}\right)^2 + \left(\frac{1}{\sqrt{2}}\right)^2} = 1$$

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

$$w_2 = \frac{1}{\sqrt{2}}$$

$$w_1x_1 + w_2x_2 + b = 0$$

when (0,1):

$$\frac{1}{\sqrt{2}}(1) + b = 0$$

$$b = \frac{-1}{\sqrt{2}}$$

when (1,0):

$$\frac{1}{\sqrt{2}}(1) + b = 0$$

$$b = \frac{-1}{\sqrt{2}}$$

$$h(x1, x2) = \begin{cases} 1, if\ (\frac{1}{\sqrt{2}})x_1 + (\frac{1}{\sqrt{2}})x_2 - \frac{1}{\sqrt{2}} \geq 0 \\ 0,\ Otherwise \end{cases}$$

```
In [ ]:
```

2. Compute the predictive labels of the following two data points: A = (3,2), B = (-1,0).

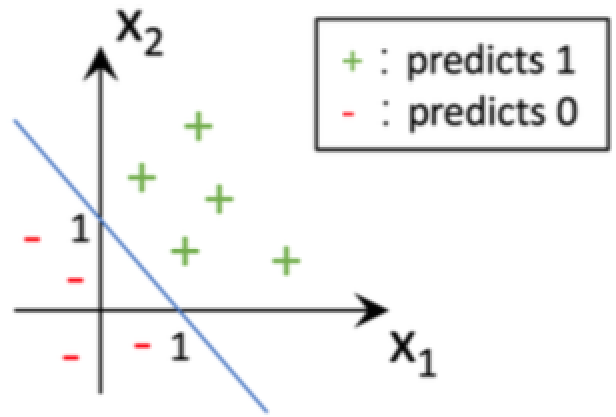


Figure 1: Decision boundary to solve the parameters.

For A:

$$
\left(\frac{1}{\sqrt{2}}\right)(3) + \left(\frac{1}{\sqrt{2}}\right)(2) - \frac{1}{\sqrt{2}} \geq 0
$$
$$
2\sqrt{2} \geq 0 \rightarrow TRUE
$$

A's predicted label is 1 this is because it will fall above the decision boundary.

For B:

$$
\left(\frac{1}{\sqrt{2}}\right)(-1) + \left(\frac{1}{\sqrt{2}}\right)(0) - \frac{1}{\sqrt{2}} \geq 0
$$
$$
-\sqrt{2} \geq 0 \rightarrow FALSE
$$

B's predicted label is 0 this is because it will fall below the decision boundary.

1.3 We are given a classifier that performs classification on R3 (the space of data points with 3 features (x1, x2, x3)) with the following decision rule:

$$h(x1, x2) = \begin{cases} 1, if\ w_1x_1 + w_2x_2 + w_3x_3 + b \geq 0 \\ 0,\ Otherwise \end{cases}$$

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

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

In addition, we set b ≤ 0 to have an unique equation for the decision boundary.

1. Compute the parameters w1, w2, w3 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 2.

Hint: Note that the normal vector is orthogonal to the hyperplane, which means the normal vector is orthogonal to any vector on the hyperplane. One way to compute the normal vector is to find two nonparallel vectors on the hyperplane, and use the orthogonal property plus the normalization constraint to solve the values for w. Or you can set the value of b as any arbitrary value and solve for w, then come back to solve the true value of b.

$$B - A = \{(-1-3), (0-2), (2-4)\} = \{-4, -2, -2\}$$
$$C - B = \{(4+1), (1-2), (5-2)\} = \{-4, -1, 3\}$$
$$(B - A) \times (C - B) = \{-4, 2, 6\}$$
$$(w_1, w_2, w_3) = \frac{\{-4, 2, 6\}}{\sqrt{(-4)^2 + (2)^2 + (6)^2}} = \{\frac{-4}{56}, \frac{2}{56}, \frac{6}{56}\}$$
$$w_1x_1 + w_2x_2 + w_3x_3 + b = 0$$
$$b = -(w_1x_1 + w_2x_2 + w_3x_3)$$
$$B = (-1, 0, 2)$$
$$b = -((\frac{-4}{56})(-1) + (\frac{2}{56})(0) + (\frac{6}{56})(2))$$
$$b = \frac{-2}{7}$$

2. Compute the predictive labels of the following three data points: p = (0,0,0), q = (1,0,5).

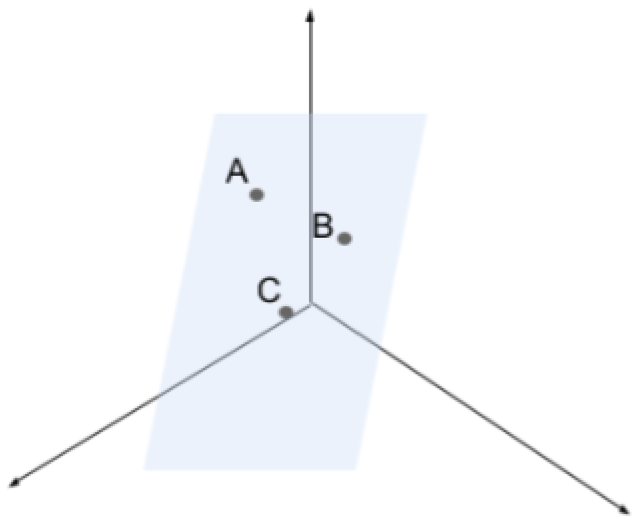


Figure 2: Decision boundary to solve the parameters.

$$p = (0,0,0)$$
$$\left[(\frac{-4}{56})(0) + (\frac{2}{56})(0) + (\frac{6}{56})(0)\right] + \frac{-2}{7} >= 0$$
$$\frac{-2}{7} >= 0 \ \rightarrow FALSE$$

So, P is < 0. P will be labeled as 0.

$$q = (1,0,5)$$
$$\left[(\frac{-4}{56})(1) + (\frac{2}{56})(0) + (\frac{6}{56})(5)\right] + \frac{-2}{7} >= 0$$
$$\frac{5}{28} >= 0 \ \rightarrow TRUE$$

So, q is > 0. q will be labeled as 1.

In [ ]:

```
```

2) Conditional Probability

Oftentimes, the performance of a binary medical diagnostic test is measured as follows:

1. True positive rate (correctly identified) = P (test + |sick+), i.e. the probability that a sick person correctly diagnosed as sick.

2. False positive rate (incorrectly identified) = P (test + |sick–), i.e. the probability that a healthy person incorrectly identified as sick.

3. True negative rate (correctly rejected) = P (test – |sick–), i.e. the probability that a healthy person correctly identified as healthy.

4. False negative rate (incorrectly rejected) = P (test – |sick+), i.e. the probability that a sick person incorrectly identified as healthy.

Here, we look at a particular mammogram tests for breast cancer. The true positive rate is 98%. The true negative rate is 94%. The incident rate of breast cancer among a certain population is 0.06%. Suppose that a person is randomly drawn from the population.

$$P(test+|sick+) = .98$$
$$P(test+|sick-) = 1 - P(test-|sick-) = 1 - .94 = 0.06$$
$$P(test-|sick-) = .94$$
$$P(sick+) = .0006$$
$$P(sick-) = .9994$$
$$P(test+) = P(test+|sick+)P(sick+) + P(test+|sick-)P(sick-) = (.98)(.0006) + (0.06)(.9994) = 0.0$$
$$P(test-) = 1 - P(test+) = 1 - 0.060552 = 0.939448$$

2.1 Given that the person is tested as positive, what is the probability of the person has breast cancer? In other words, what is P (cancer + |test+)? (the cancer+ means sick+ in slides and previous page)

$$P(cancer+|test+) = P(sick+|test+) = \frac{P(test+|sick+)P(sick+)}{P(test+)} = \frac{(.98)(.0006)}{(0.060552)}$$
$$= 0.0097106619104240982956797463337296868806975822433610780 81$$

2.2 Given that the person is tested as negative, what is the probability of the person does not has breast cancer? In other words, what is P (cancer – |test–)?

$$P(cancer-|test-) = P(sick-|test-) = \frac{P(test-|sick-)P(sick-)}{P(test-)} = \frac{(.94)(.9994)}{(0.939448)}$$
$$= 0.99998722654154354471987805604993570692576917509005288211 8$$

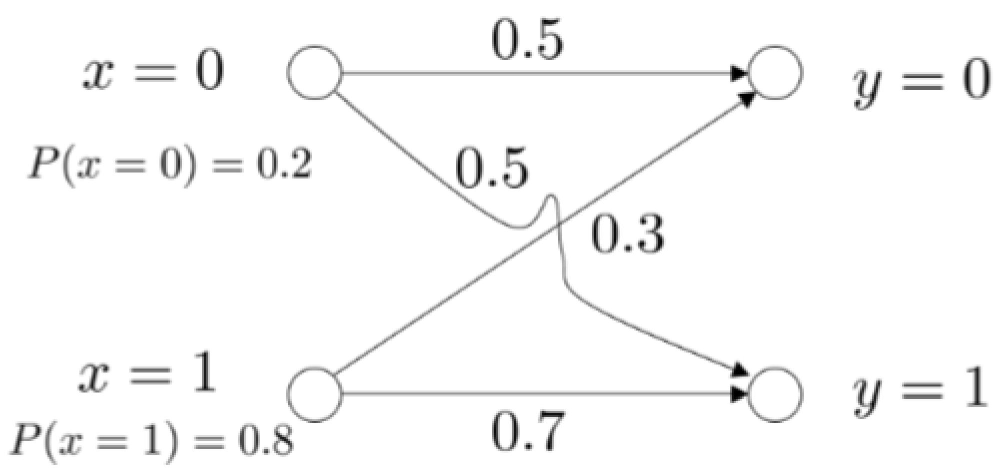
2.3 Compute precision, recall, and $F - value = \frac{2\times precision\times recall}{precision+recall}$

$$precision = \frac{P(sick+\cap test+)}{P(test+)} = P(sick+|test+) = \frac{P(test+|sick+)P(sick+)}{P(test+)} = \frac{(.98)(.0006)}{(0.060552)}$$
$$= 0.0097106619104240982956797463337296868806975822433610780 81$$
$$recall = \frac{P(sick+\cap test+)}{P(sick+)} = P(test+|sick+) = .98$$
$$F - value = \frac{2\times precision\times recall}{precision+recall} = \frac{2\times .00971066191042409829567974633372968688069758224 3}{.00971066191042409829567974633372968688069758224336}$$
$$= 0.0192307692307692307692307692307692307692307692307692307 67$$

I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here.I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here.

3) Binary Communication System

For the binary communication system shown below, compute the following probabilities:



The parameter p_e, probability of error.

The parameter $1 - p_e$, probability of no error.

a) P(x=2)

x=2 is not a valid option since because it is not 0 or 1 (We are using a binary coding scheme) So, this is not a valid question.

b) P(y=0|x=1)

"given that you sent a 1, whats the probability that they recieved a 0?"

$$P(y = 0|x = 1) = .3$$

c) P(y=0)

$$P(y = 0) = P(y = 0|x = 1)P(x = 1) + P(y = 0|x = 0)P(x = 0) = (.3)(.8) + (.5)(.2) = 0.34$$

d) P(x=1|y=0)

"given that they recieved a 0, whats the probability that you sent a 1?"

$$P(x = 1|y = 0) = \frac{P(y = 0|x = 1)P(x = 1)}{P(y = 0)} = \frac{(.3)(.8)}{0.34} = 0.705882$$

4) Minimizers and Maximizers

4.1 Probability

The joint probability mass function of the random variables (x, y) is given by the following table. Compute the following:

	$x=0$	$x=1$	$x=2$
$y=0$	0.2	0.1	0.1
$y=1$	0.3	0.2	0.1

1. (i∗,j∗) = argmax(i,j) P(x = i,y = j)

$$(i*, j*) = argmax(i, j)P(x = i, y = j) = argmax(x = 2, y = 1)P(x = 2 \cap y = 1) = 0.1$$

2. j∗ =argminjP(y=j|x=1)

$$j* = argminjP(y = j|x = 1) = argmin(0)P(y = 0|x = 1) = \frac{P(y = 0 \cap x = 1)}{P(x = 1)} = \frac{(.1)}{(.1 + .2)}$$
$$= 0.3333333333333333333333333333333333333333333333333333333$$

3. i∗ =argmaxiP(x=i)

$$i* = argmaxiP(x = i) = argmax(2)P(x = 2) = .1 + .1 = 0.2$$

I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here. I dont know why it cuts off here.I dont know why it cuts off here.

4.2 Function

(Check Lecture Slide 4, Page44-48) An unknown estimator is given an estimation problem to find the maximizer of the objective function G(θ) ∈ (0, 2]:

$$\theta = argmax_{\theta}G(\theta)$$

The solution to Eq. 1 by the estimator is θ1 = 67. Given this information, obtain θ∗ such that

$$\theta^* = argmin_{\theta}[10 - 3ln(G(\theta))]$$
$$ln[G(\theta)]\ is\ an\ increasing$$
$$-ln[G(\theta)]\ is\ not\ increasing$$
$$-3ln[G(\theta)]\ is\ not\ increasing\ with\ scalar\ 3$$
$$G(\theta)\ is\ a\ maximum$$
$$so,\ \theta^* = 67$$

5) Training vs. Testing Errors

In [41]:
```
%config InlineBackend.figure_format = 'retina'
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
```

In [76]:
```
# Iris dataset.
iris = datasets.load_iris()     # Load Iris dataset.

X = iris.data                   # The shape of X is (150, 4), which mean
s
                                # there are 150 data points, each data p
oint
                                # has 4 features.
print(X)
# Here for convenience, we divide the 3 kinds of flowers into 2 groups:
#     Y = 0 (or False):  Setosa (original value 0) / Versicolor (origina
l 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 (V
irginica)
#    Y[i] = False (which is equivalent to 0) if iris.target[k] <= 1.5 (S
etosa / Versicolor)

Y = (iris.target > 1.5).reshape(-1,1) # The shape of Y is (150, 1), whic
h means
                                # there are 150 data points, each data p
oint
                                # 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 arr
ay

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

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.  3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.1 1.5 0.1]
 [4.4 3.  1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.  3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.  3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.  1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.  3.3 1.4 0.2]
 [7.  3.2 4.7 1.4]
 [6.4 3.2 4.5 1.5]
 [6.9 3.1 4.9 1.5]
 [5.5 2.3 4.  1.3]
 [6.5 2.8 4.6 1.5]
 [5.7 2.8 4.5 1.3]
 [6.3 3.3 4.7 1.6]
```

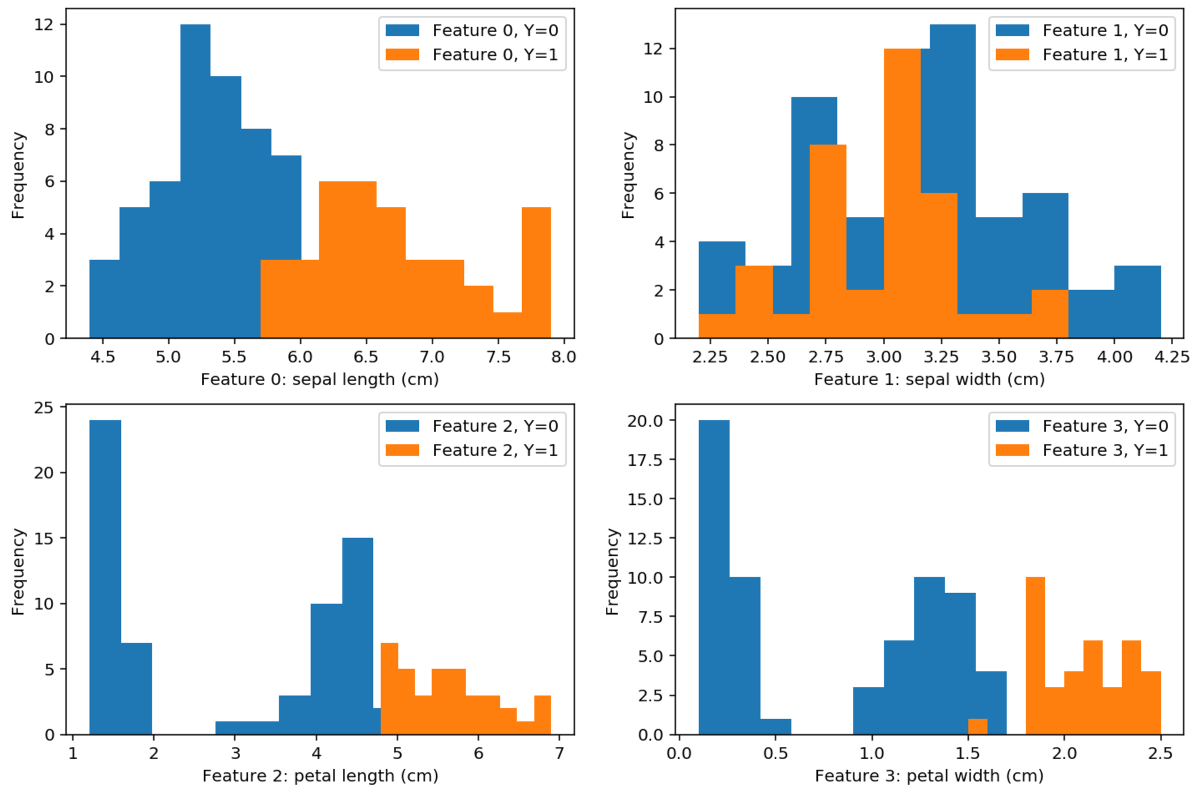
```
 [4.9 2.4 3.3 1. ]
 [6.6 2.9 4.6 1.3]
 [5.2 2.7 3.9 1.4]
 [5.  2.  3.5 1. ]
 [5.9 3.  4.2 1.5]
 [6.  2.2 4.  1. ]
 [6.1 2.9 4.7 1.4]
 [5.6 2.9 3.6 1.3]
 [6.7 3.1 4.4 1.4]
 [5.6 3.  4.5 1.5]
 [5.8 2.7 4.1 1. ]
 [6.2 2.2 4.5 1.5]
 [5.6 2.5 3.9 1.1]
 [5.9 3.2 4.8 1.8]
 [6.1 2.8 4.  1.3]
 [6.3 2.5 4.9 1.5]
 [6.1 2.8 4.7 1.2]
 [6.4 2.9 4.3 1.3]
 [6.6 3.  4.4 1.4]
 [6.8 2.8 4.8 1.4]
 [6.7 3.  5.  1.7]
 [6.  2.9 4.5 1.5]
 [5.7 2.6 3.5 1. ]
 [5.5 2.4 3.8 1.1]
 [5.5 2.4 3.7 1. ]
 [5.8 2.7 3.9 1.2]
 [6.  2.7 5.1 1.6]
 [5.4 3.  4.5 1.5]
 [6.  3.4 4.5 1.6]
 [6.7 3.1 4.7 1.5]
 [6.3 2.3 4.4 1.3]
 [5.6 3.  4.1 1.3]
 [5.5 2.5 4.  1.3]
 [5.5 2.6 4.4 1.2]
 [6.1 3.  4.6 1.4]
 [5.8 2.6 4.  1.2]
 [5.  2.3 3.3 1. ]
 [5.6 2.7 4.2 1.3]
 [5.7 3.  4.2 1.2]
 [5.7 2.9 4.2 1.3]
 [6.2 2.9 4.3 1.3]
 [5.1 2.5 3.  1.1]
 [5.7 2.8 4.1 1.3]
 [6.3 3.3 6.  2.5]
 [5.8 2.7 5.1 1.9]
 [7.1 3.  5.9 2.1]
 [6.3 2.9 5.6 1.8]
 [6.5 3.  5.8 2.2]
 [7.6 3.  6.6 2.1]
 [4.9 2.5 4.5 1.7]
 [7.3 2.9 6.3 1.8]
 [6.7 2.5 5.8 1.8]
 [7.2 3.6 6.1 2.5]
 [6.5 3.2 5.1 2. ]
 [6.4 2.7 5.3 1.9]
 [6.8 3.  5.5 2.1]
 [5.7 2.5 5.  2. ]
```

```
 [5.8 2.8 5.1 2.4]
 [6.4 3.2 5.3 2.3]
 [6.5 3.  5.5 1.8]
 [7.7 3.8 6.7 2.2]
 [7.7 2.6 6.9 2.3]
 [6.  2.2 5.  1.5]
 [6.9 3.2 5.7 2.3]
 [5.6 2.8 4.9 2. ]
 [7.7 2.8 6.7 2. ]
 [6.3 2.7 4.9 1.8]
 [6.7 3.3 5.7 2.1]
 [7.2 3.2 6.  1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3.  4.9 1.8]
 [6.4 2.8 5.6 2.1]
 [7.2 3.  5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2. ]
 [6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
 [6.1 2.6 5.6 1.4]
 [7.7 3.  6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
(150, 4)
(150, 1)
[5.8 4.  1.2 0.2 0. ]
```

In [77]:
```
# 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 [48]:
```
# Show the histograms of each feature.
plt.figure(figsize=(12,8))
for j in range(4):
    Xj_train = X_train[:,j]
    Xj_when_Y0_train = [Xj_train[i] for i in range(len(Xj_train)) if Y_t
rain[i] == 0]
    Xj_when_Y1_train = [Xj_train[i] for i in range(len(Xj_train)) if Y_t
rain[i] == 1]

    plt.subplot(2, 2, j+1)
    plt.hist(Xj_when_Y0_train, label='Feature {}, Y=0'.format(j))
    plt.hist(Xj_when_Y1_train, label='Feature {}, Y=1'.format(j))
    plt.xlabel('Feature {}: {}'.format(j, iris.feature_names[j]))
    plt.ylabel('Frequency')
    plt.legend()
plt.show()
```



In [49]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score

reg = LinearRegression().fit(X_train, Y_train)
print(reg.coef_)
print(reg.intercept_)

w = reg.coef_
b = reg.intercept_

y_hat_train = np.dot(X_train, w) + b
label_train = (y_hat_train <= .5)
label_train = label_train*1

epilson_i_train = y_hat_train - Y_train
epilson_i_train_sqrted = np.square(epilson_i_train)

y_hat_test = np.dot(X_test, w) + b
epilson_i_test = y_hat_test - Y_test
epilson_i_test_sqrted = np.square(epilson_i_test)
```

```
[ 0.12975624  0.12249935 -0.11714156  0.67102651]
-1.1698768088050127
```

Training error of the regression model.

In [50]:
```
#Training error of the regression model
sum_epilson_i_train_sqrted = np.sum(epilson_i_train_sqrted)
regression_training_error = np.sqrt(sum_epilson_i_train_sqrted/len(epils
on_i_train_sqrted))
print(regression_training_error)
```

```
0.27976412743241214
```

Testing error of the regression model.

In [51]:
```
#Testing error of the regression model.
sum_epilson_i_test_sqrted = np.sum(epilson_i_test_sqrted)
regression_test_error = np.sqrt(sum_epilson_i_test_sqrted/len(epilson_i_
test_sqrted))
print(regression_test_error)
```

```
0.3310071344139558
```

Training error of the classification model.

In [52]:
```
#Training error of the classification model.
list = (label_train == Y_train)
list = 1*list
training_classification__error = sum(list) / len(list)
print(training_classification__error)
```

```
0.06
```

Testing error of the classification model.

In [53]:
```
#Testing error of the classification model.
label_test = (y_hat_test <= .5)
label_test = label_test*1
list_test = label_test == Y_test
test_classification__error = sum(list_test) / len(list_test)
print(test_classification__error)
```

```
0.14
```

6) Decision Stump

In this problem, we will perform a binary classification task on the Iris dataset. Again, this dataset has 150 data points, where each data point x ∈ R4 has 4 features and its corresponding label y ∈ {0, 1}. To classify these 2 labels above, we decide to utilize a decision stump. The decision stump works as follows (for simplicity, we restrict our attention to uni-directional decision stumps):

• Given the j-th feature xi(j) and a threshold Thj, for data point i, the classification function is defined by $y = f(x, j, Th_j)$**as:**

$$f(x, j, Th_j) = \begin{cases} 1, if\ x(j) > Thj \\ 0,\ Otherwise \end{cases}$$

Based on the decision stump above, we wish to write an algorithm to find the best feature and best threshold on training set to create a "best" decision stump, in a sense that such decision stump achieves the highest accuracy on training set.

Follow the instructions in the skeleton code and report:

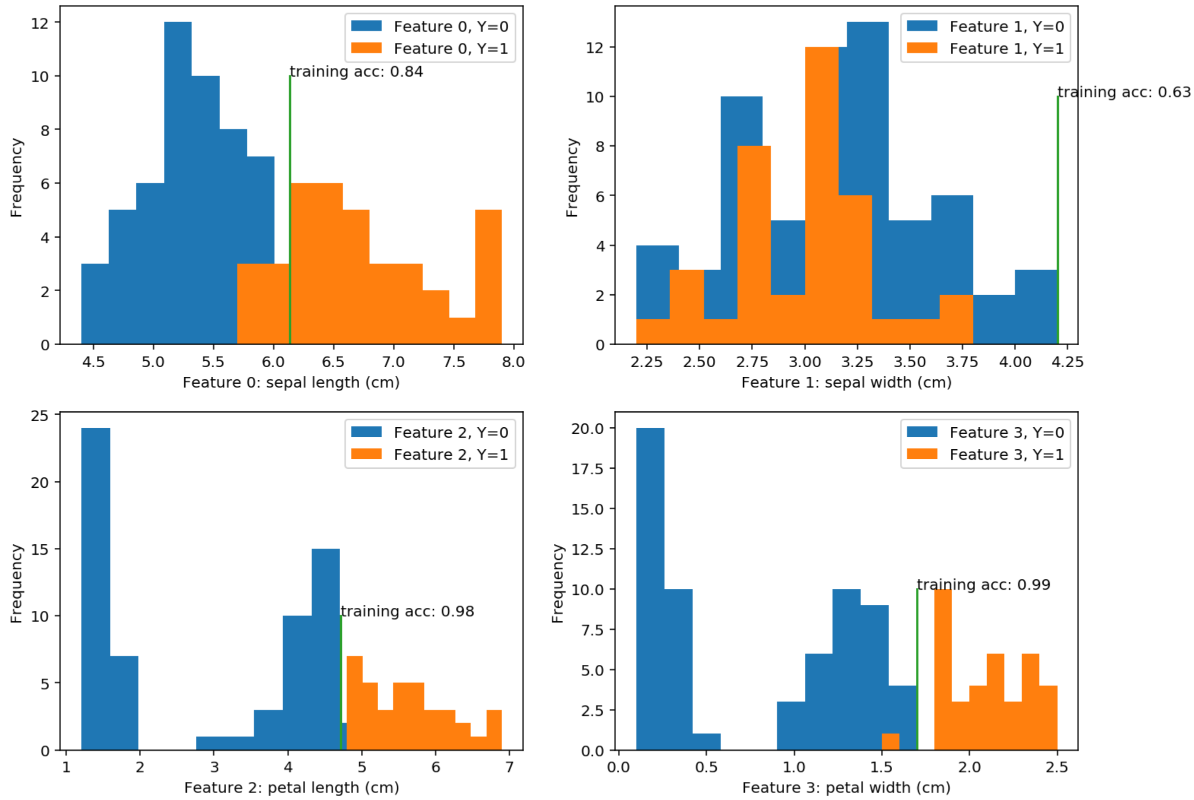
In [36]:

```
# Calculate the accuracy of prediction given feature, target and thresho
ld.
def calc_acc(Xj, Y, thres):
    """
    Calculate the accuracy given feature, target and threshold.
        Xj:    j-th feature. This array only contains 1 feature for all
 data points,
               so the shape should be (count of data points,)
        Y:     Target array. Shape: (count of data points,)
        thres: Threshold.
    Return the accuracy of prediction.
    """

    # Step 1. Count the number of correct predictions and incorrect pred
ictions.
    #         Here, for simplicity, we assume:
    #              If feature <= threshold, we predict it as Y = 0.
    #              If feature  > threshold, we predict it as Y = 1.
    n_correct = 0
    n_incorrect = 0


    # ******** To be filled ********
    thres_arr = np.full((Xj.shape[0], ), thres)
    label = (Xj > thres_arr)
    label =1*label

    correct = 1*(label == Y)
    n_correct = sum(correct)
    n_incorrect = len(Y) - n_correct

    # Step 2. Calculate the accuracy.
    acc = 1.0 * n_correct / (n_correct + n_incorrect)

    return acc
```

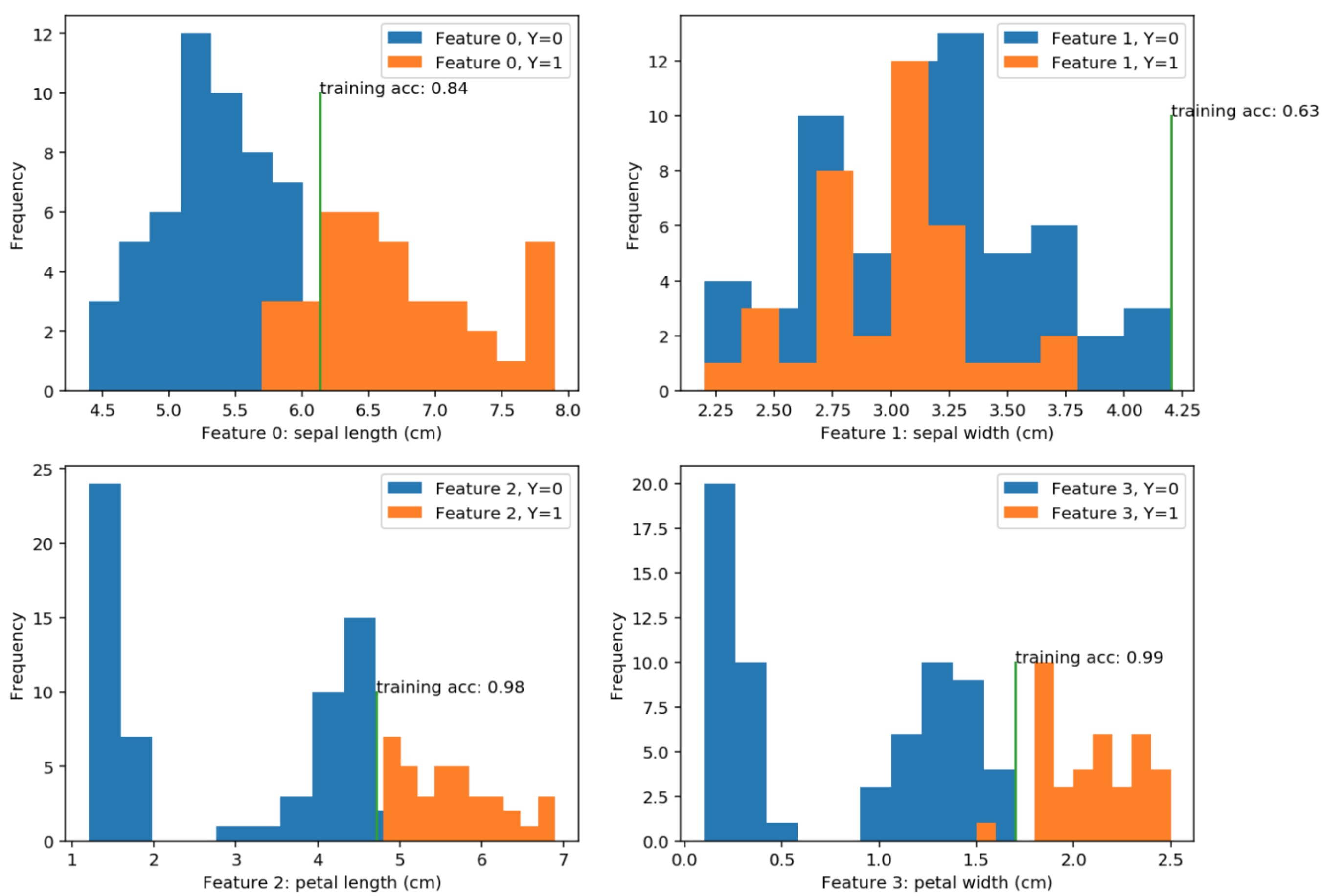
In [37]:
```
# Show the histograms of each feature.
plt.figure(figsize=(12,9))

all_max_acc = 0.0  # Max training accuracy among all features.
all_thres   = None # Threshold when reach the max training accuracy.
all_feature = None # Index of feature when reach the max training accura
cy.

print(X_train)
# Loop over 4 features. j: index of current feature.
for j in range(4):
    # Get data.
    Xj_train = X_train[:,j] # Array of feature j.
    Xj_when_Y0_train = [Xj_train[i] for i in range(len(Xj_train)) if Y_t
rain[i] == 0] # Array of feature j when Y = 0.
    Xj_when_Y1_train = [Xj_train[i] for i in range(len(Xj_train)) if Y_t
rain[i] == 1] # Array of feature j when Y = 1.

    current_max_acc = 0.0    # Max training accuracy in current feature.
    current_thres   = None   # Threshold when reach the max accuracy in
 current feature.

    # Loop over all possible values for threshold. Here we consider 100
 numbers between min and max of current feature.
    for thres in np.linspace(Xj_train.min(), Xj_train.max(), 100):
        # Calculate the accuracy on training data given feature, target
 and threshold.
        acc = calc_acc(Xj_train, Y_train, thres)
        # Update the current max accuracy if possible.
        if acc > current_max_acc:
            current_max_acc = acc
            current_thres = thres

    # Update the max training accuracy among all features if possible.
    if current_max_acc > all_max_acc:
        all_max_acc = current_max_acc
        all_thres   = current_thres
        all_feature = j

    # Plot the histograms and the best decision stump in current featur
e.
    plt.subplot(2, 2, j+1)
    plt.hist(Xj_when_Y0_train, label='Feature {}, Y=0'.format(j))
    plt.hist(Xj_when_Y1_train, label='Feature {}, Y=1'.format(j))
    plt.plot([current_thres, current_thres], [0, 10])
    plt.text(current_thres, 10, 'training acc: {}'.format(current_max_ac
c))
    plt.xlabel('Feature {}: {}'.format(j, iris.feature_names[j]))
    plt.ylabel('Frequency')
    plt.legend()
plt.show()
```

```
[[5.8 4.  1.2 0.2]
 [5.1 2.5 3.  1.1]
 [6.6 3.  4.4 1.4]
 [5.4 3.9 1.3 0.4]
 [7.9 3.8 6.4 2. ]
 [6.3 3.3 4.7 1.6]
 [6.9 3.1 5.1 2.3]
 [5.1 3.8 1.9 0.4]
 [4.7 3.2 1.6 0.2]
 [6.9 3.2 5.7 2.3]
 [5.6 2.7 4.2 1.3]
 [5.4 3.9 1.7 0.4]
 [7.1 3.  5.9 2.1]
 [6.4 3.2 4.5 1.5]
 [6.  2.9 4.5 1.5]
 [4.4 3.2 1.3 0.2]
 [5.8 2.6 4.  1.2]
 [5.6 3.  4.5 1.5]
 [5.4 3.4 1.5 0.4]
 [5.  3.2 1.2 0.2]
 [5.5 2.6 4.4 1.2]
 [5.4 3.  4.5 1.5]
 [6.7 3.  5.  1.7]
 [5.  3.5 1.3 0.3]
 [7.2 3.2 6.  1.8]
 [5.7 2.8 4.1 1.3]
 [5.5 4.2 1.4 0.2]
 [5.1 3.8 1.5 0.3]
 [6.1 2.8 4.7 1.2]
 [6.3 2.5 5.  1.9]
 [6.1 3.  4.6 1.4]
 [7.7 3.  6.1 2.3]
 [5.6 2.5 3.9 1.1]
 [6.4 2.8 5.6 2.1]
 [5.8 2.8 5.1 2.4]
 [5.3 3.7 1.5 0.2]
 [5.5 2.3 4.  1.3]
 [5.2 3.4 1.4 0.2]
 [6.5 2.8 4.6 1.5]
 [6.7 2.5 5.8 1.8]
 [6.8 3.  5.5 2.1]
 [5.1 3.5 1.4 0.3]
 [6.  2.2 5.  1.5]
 [6.3 2.9 5.6 1.8]
 [6.6 2.9 4.6 1.3]
 [7.7 2.6 6.9 2.3]
 [5.7 3.8 1.7 0.3]
 [5.  3.6 1.4 0.2]
 [4.8 3.  1.4 0.3]
 [5.2 2.7 3.9 1.4]
 [5.1 3.4 1.5 0.2]
 [5.5 3.5 1.3 0.2]
 [7.7 3.8 6.7 2.2]
 [6.9 3.1 5.4 2.1]
 [7.3 2.9 6.3 1.8]
 [6.4 2.8 5.6 2.2]
 [6.2 2.8 4.8 1.8]
```

```
 [6.  3.4 4.5 1.6]
 [7.7 2.8 6.7 2. ]
 [5.7 3.  4.2 1.2]
 [4.8 3.4 1.6 0.2]
 [5.7 2.5 5.  2. ]
 [6.3 2.7 4.9 1.8]
 [4.8 3.  1.4 0.1]
 [4.7 3.2 1.3 0.2]
 [6.5 3.  5.8 2.2]
 [4.6 3.4 1.4 0.3]
 [6.1 3.  4.9 1.8]
 [6.5 3.2 5.1 2. ]
 [6.7 3.1 4.4 1.4]
 [5.7 2.8 4.5 1.3]
 [6.7 3.3 5.7 2.5]
 [6.  3.  4.8 1.8]
 [5.1 3.8 1.6 0.2]
 [6.  2.2 4.  1. ]
 [6.4 2.9 4.3 1.3]
 [6.5 3.  5.5 1.8]
 [5.  2.3 3.3 1. ]
 [6.3 3.3 6.  2.5]
 [5.5 2.5 4.  1.3]
 [5.4 3.7 1.5 0.2]
 [4.9 3.1 1.5 0.1]
 [5.2 4.1 1.5 0.1]
 [6.7 3.3 5.7 2.1]
 [4.4 3.  1.3 0.2]
 [6.  2.7 5.1 1.6]
 [6.4 2.7 5.3 1.9]
 [5.9 3.  5.1 1.8]
 [5.2 3.5 1.5 0.2]
 [5.1 3.3 1.7 0.5]
 [5.8 2.7 4.1 1. ]
 [4.9 3.1 1.5 0.1]
 [7.4 2.8 6.1 1.9]
 [6.2 2.9 4.3 1.3]
 [7.6 3.  6.6 2.1]
 [6.7 3.  5.2 2.3]
 [6.3 2.3 4.4 1.3]
 [6.2 3.4 5.4 2.3]
 [7.2 3.6 6.1 2.5]
 [5.6 2.9 3.6 1.3]]
```



- **All 4 histograms in last part of the code.**



- **The best feature, best threshold, training and test accuracy in last part of the code.**

In [39]:
```
# Use the best feature and best threshold on test set.

Xj_test = X_test[:, all_feature] # Array of best feature.
test_acc = calc_acc(Xj_test, Y_test, all_thres)
print('Best feature: {}'.format(all_feature))
print('Best threshold: {:.2f}'.format(all_thres))
print('Training accuracy of best feature: {:.2f}'.format(all_max_acc))
print('Test accuracy of best feature: {:.2f}'.format(test_acc))
```

```
Best feature: 3
Best threshold: 1.70
Training accuracy of best feature: 0.99
Test accuracy of best feature: 0.90
```