Khoi Duong

Prof. Yang

CS483

6/25/2022

HW#2

1.

Hypothesis: $h_0 = \theta_0 + \theta_1 x_1$

Cost function: $J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}[(\theta_0 + \theta_1 x_1^{(i)}) - y^{(i)}]^2$

Gradient decent algorithm:

$$\theta_0 = \theta_0 - \alpha\frac{\partial J(\theta)}{\partial \theta_0}$$

$$\theta_1 = \theta_1 - \alpha\frac{\partial J(\theta)}{\partial \theta_1}$$

We have:

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}[(\theta_0 + \theta_1 x_1^{(i)}) - y^{(i)}]$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m}[(\theta_0 + \theta_1 x_1^{(i)}) - y^{(i)}] * x_1^{(i)}$$

Let $\theta_0 = \theta_1 = 0$, m = 14, and learning rate $\alpha = 0.01$

We have the source code below:

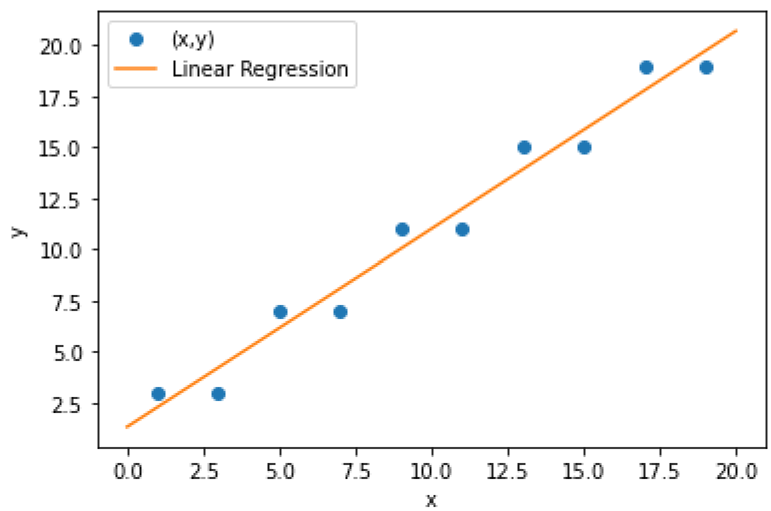
```
import numpy as np
import matplotlib.pyplot as plt
```

```
x = [1,3,5,7,9,11,13,15,17,19]
y = [3,3,7,7,11,11,15,15,19,19]

theta_0 = theta_1 = 0
alpha = 0.0001
i = 0

while i <= 500000:
  diff_theta_0 = diff_theta_1 = 0
  for m in range (10):
    diff_theta_0 += theta_0 + theta_1*x[m]  - y[m]
    diff_theta_1 += (theta_0 + theta_1*x[m] - y[m]) * x[m]
  diff_theta_0 = diff_theta_0 * (1/10)
  diff_theta_1 = diff_theta_1 * (1/10)

  theta_0 = theta_0 - alpha * diff_theta_0
  theta_1 = theta_1 - alpha * diff_theta_1
  i += 1

print("diff_theta_0 = " + str(diff_theta_0) + ", " + "Theta 0 = ",
str(theta_0))
print("diff_theta_1 = " + str(diff_theta_1) + ", " + "Theta 1 = ",
str(theta_1))

# Plot (x,y) points and fitting curve
x_fit = np.linspace(0, 20, 100)
y_fit = theta_0 + theta_1*x_fit
plt.plot(x, y, 'o', x_fit, y_fit)

# Label the axes and put the legend in the graph
plt.xlabel('x')
plt.ylabel('y')
plt.legend(['(x,y)', 'Linear Regression'])
plt.show()
```

Run program & result:

```
diff_theta_0 = -1.3245418937213316e-06, Theta 0 =  1.303024934622789

diff_theta_1 = 9.977470973865366e-08, Theta 1 =  0.9696973740867997
```

We have the graph below:



Thus, the linear regression line is y = 0.9697x + 1.3030

2.

For binary classification, we have to use the sigmoid function (logistic regression)

We have the hypothesis function with 9 features:

$$h_\theta(x) = g(\theta_0 + \theta_1x_1 + \theta_2x_2 + \theta_3x_3 + \theta_4x_4 + \theta_5x_5 + \theta_6x_6 + \theta_7x_7 + \theta_8x_8 + \theta_9x_9)$$

Where $g(z) = \dfrac{e^z}{1 + e^z}$

In this data table, we will re-value class benign as 0 (change from 2 to 0), and class malignant as 1 (change from 4 to 1).

We have the cost function as below:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} * \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) * \log(h_\theta(x^{(i)}))]$$

And we also have the loss function:

$$\frac{\partial}{\partial\theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) * x_j^{(i)}$$

Let $\theta_0 = \theta_1 = \theta_2 = \theta_3 = \theta_4 = \theta_5 = \theta_6 = \theta_7 = \theta_8 = \theta_9 = 0$, $m = 32$, and learning rate $\alpha = 0.0001$

We have the source code below:

```
import numpy as np
import matplotlib.pyplot as plt
import math

clump_thickness =
[8,5,1,8,7,4,4,10,6,7,10,3,8,1,5,3,5,2,1,3,2,10,2,3,2,10,6,5,2,10,6,5]
uni_cell_size =
[10,3,1,7,4,1,1,7,1,3,5,1,4,1,2,2,1,1,1,1,1,7,1,1,1,10,2,4,5,4,10,6]
uni_cell_shape =
[10,3,1,5,6,1,1,7,1,2,5,1,5,1,3,1,1,1,3,1,1,7,1,2,1,10,1,4,3,3,10,5]
marginal_adhesion =
[8,3,1,10,4,1,1,6,1,10,3,1,1,1,4,1,1,1,1,1,1,3,2,1,1,8,1,9,3,1,2,6]
sing_epi_cell_size =
[7,2,2,7,6,2,2,4,2,5,6,2,2,2,2,1,2,2,2,1,2,8,2,2,2,6,1,2,6,3,8,10]
bare_nuclei =
[10,3,3,9,1,1,1,10,1,10,7,1,5,1,7,1,1,1,1,1,1,5,1,1,1,1,1,10,7,3,10,1]
bland_chromatin =
[9,4,3,5,4,2,3,4,3,5,7,2,7,3,3,2,2,2,1,2,3,7,3,2,2,8,7,5,7,6,7,3]
normal_nucleoli =
[7,4,1,5,3,1,1,1,1,4,10,1,3,1,6,1,1,1,1,1,1,4,1,1,1,9,1,6,5,5,3,1]
mitoses =
[1,1,1,4,1,1,1,2,1,4,1,1,1,1,1,1,1,1,1,1,1,3,1,1,1,1,1,1,1,2,3,1]
y = [1,1,0,1,1,0,0,1,0,1,1,0,1,0,1,0,0,0,0,0,0,1,0,0,0,1,0,1,1,1,1,1]

theta_0 = theta_1 = theta_2 = theta_3 = theta_4 = theta_5 = theta_6 = theta_7
= theta_8 = theta_9 = 0
alpha = 0.0001
i = 0

while i <= 500000:
```

```
  diff_theta_0 = diff_theta_1 = diff_theta_2 = diff_theta_3 = diff_theta_4
= diff_theta_5 = diff_theta_6 = diff_theta_7 = diff_theta_8 = diff_theta_9
= 0
  for m in range (32):
    diff_theta_0 += (1/(1 + math.exp(-(theta_0 + theta_1*clump_thickness[m]
+ theta_2*uni_cell_size[m] + theta_3*uni_cell_shape[m] +
theta_4*marginal_adhesion[m] + theta_5*sing_epi_cell_size[m] +
theta_6*bare_nuclei[m] + theta_7*bland_chromatin[m] +
theta_8*normal_nucleoli[m] + theta_9*mitoses[m])))) - y[m]
    diff_theta_1 += ((1/(1 + math.exp(-(theta_0 +
theta_1*clump_thickness[m] + theta_2*uni_cell_size[m] +
theta_3*uni_cell_shape[m] + theta_4*marginal_adhesion[m] +
theta_5*sing_epi_cell_size[m] + theta_6*bare_nuclei[m] +
theta_7*bland_chromatin[m] + theta_8*normal_nucleoli[m] +
theta_9*mitoses[m])))) - y[m]) * clump_thickness[m]
    diff_theta_2 += ((1/(1 + math.exp(-(theta_0 +
theta_1*clump_thickness[m] + theta_2*uni_cell_size[m] +
theta_3*uni_cell_shape[m] + theta_4*marginal_adhesion[m] +
theta_5*sing_epi_cell_size[m] + theta_6*bare_nuclei[m] +
theta_7*bland_chromatin[m] + theta_8*normal_nucleoli[m] +
theta_9*mitoses[m])))) - y[m]) * uni_cell_size[m]
    diff_theta_3 += ((1/(1 + math.exp(-(theta_0 +
theta_1*clump_thickness[m] + theta_2*uni_cell_size[m] +
theta_3*uni_cell_shape[m] + theta_4*marginal_adhesion[m] +
theta_5*sing_epi_cell_size[m] + theta_6*bare_nuclei[m] +
theta_7*bland_chromatin[m] + theta_8*normal_nucleoli[m] +
theta_9*mitoses[m])))) - y[m]) * uni_cell_shape[m]
    diff_theta_4 += ((1/(1 + math.exp(-(theta_0 +
theta_1*clump_thickness[m] + theta_2*uni_cell_size[m] +
theta_3*uni_cell_shape[m] + theta_4*marginal_adhesion[m] +
theta_5*sing_epi_cell_size[m] + theta_6*bare_nuclei[m] +
theta_7*bland_chromatin[m] + theta_8*normal_nucleoli[m] +
theta_9*mitoses[m])))) - y[m]) * marginal_adhesion[m]
    diff_theta_5 += ((1/(1 + math.exp(-(theta_0 +
theta_1*clump_thickness[m] + theta_2*uni_cell_size[m] +
theta_3*uni_cell_shape[m] + theta_4*marginal_adhesion[m] +
theta_5*sing_epi_cell_size[m] + theta_6*bare_nuclei[m] +
theta_7*bland_chromatin[m] + theta_8*normal_nucleoli[m] +
theta_9*mitoses[m])))) - y[m]) * sing_epi_cell_size[m]
```

```
    diff_theta_6 += ((1/(1 + math.exp(-(theta_0 +
theta_1*clump_thickness[m] + theta_2*uni_cell_size[m] +
theta_3*uni_cell_shape[m] + theta_4*marginal_adhesion[m] +
theta_5*sing_epi_cell_size[m] + theta_6*bare_nuclei[m] +
theta_7*bland_chromatin[m] + theta_8*normal_nucleoli[m] +
theta_9*mitoses[m])))) - y[m]) * bare_nuclei[m]
    diff_theta_7 += ((1/(1 + math.exp(-(theta_0 +
theta_1*clump_thickness[m] + theta_2*uni_cell_size[m] +
theta_3*uni_cell_shape[m] + theta_4*marginal_adhesion[m] +
theta_5*sing_epi_cell_size[m] + theta_6*bare_nuclei[m] +
theta_7*bland_chromatin[m] + theta_8*normal_nucleoli[m] +
theta_9*mitoses[m])))) - y[m]) * bland_chromatin[m]
    diff_theta_8 += ((1/(1 + math.exp(-(theta_0 +
theta_1*clump_thickness[m] + theta_2*uni_cell_size[m] +
theta_3*uni_cell_shape[m] + theta_4*marginal_adhesion[m] +
theta_5*sing_epi_cell_size[m] + theta_6*bare_nuclei[m] +
theta_7*bland_chromatin[m] + theta_8*normal_nucleoli[m] +
theta_9*mitoses[m])))) - y[m]) * normal_nucleoli[m]
    diff_theta_9 += ((1/(1 + math.exp(-(theta_0 +
theta_1*clump_thickness[m] + theta_2*uni_cell_size[m] +
theta_3*uni_cell_shape[m] + theta_4*marginal_adhesion[m] +
theta_5*sing_epi_cell_size[m] + theta_6*bare_nuclei[m] +
theta_7*bland_chromatin[m] + theta_8*normal_nucleoli[m] +
theta_9*mitoses[m])))) - y[m]) * mitoses[m]
  diff_theta_0 = diff_theta_0 * (1/32)
  diff_theta_1 = diff_theta_1 * (1/32)
  diff_theta_2 = diff_theta_2 * (1/32)
  diff_theta_3 = diff_theta_3 * (1/32)
  diff_theta_4 = diff_theta_4 * (1/32)
  diff_theta_5 = diff_theta_5 * (1/32)
  diff_theta_6 = diff_theta_6 * (1/32)
  diff_theta_7 = diff_theta_7 * (1/32)
  diff_theta_8 = diff_theta_8 * (1/32)
  diff_theta_9 = diff_theta_9 * (1/32)

  theta_0 = theta_0 - alpha * diff_theta_0
  theta_1 = theta_1 - alpha * diff_theta_1
  theta_2 = theta_2 - alpha * diff_theta_2
  theta_3 = theta_3 - alpha * diff_theta_3
  theta_4 = theta_4 - alpha * diff_theta_4
```

```
    theta_5 = theta_5 - alpha * diff_theta_5
    theta_6 = theta_6 - alpha * diff_theta_6
    theta_7 = theta_7 - alpha * diff_theta_7
    theta_8 = theta_8 - alpha * diff_theta_8
    theta_9 = theta_9 - alpha * diff_theta_9
    i += 1

print("diff_theta_0 = " + str(diff_theta_0) + ", " + "Theta 0 = ", str(theta_0))
print("diff_theta_1 = " + str(diff_theta_1) + ", " + "Theta 1 = ", str(theta_1))
print("diff_theta_2 = " + str(diff_theta_2) + ", " + "Theta 2 = ", str(theta_2))
print("diff_theta_3 = " + str(diff_theta_3) + ", " + "Theta 3 = ", str(theta_3))
print("diff_theta_4 = " + str(diff_theta_4) + ", " + "Theta 4 = ", str(theta_4))
print("diff_theta_5 = " + str(diff_theta_5) + ", " + "Theta 5 = ", str(theta_5))
print("diff_theta_6 = " + str(diff_theta_6) + ", " + "Theta 6 = ", str(theta_6))
print("diff_theta_7 = " + str(diff_theta_7) + ", " + "Theta 7 = ", str(theta_7))
print("diff_theta_8 = " + str(diff_theta_8) + ", " + "Theta 8 = ", str(theta_8))
print("diff_theta_9 = " + str(diff_theta_9) + ", " + "Theta 9 = ", str(theta_9))

# predict which class of last two records
data_pred_1 = [10, 10, 10, 4, 8, 1, 8, 10, 1]
data_pred_2 = [6, 6, 6, 9, 6, 2, 7, 8, 1]
y_pred_1 = (1/(1 + math.exp(-(theta_0 + theta_1*data_pred_1[0] + theta_2*data_pred_1[1] + theta_3*data_pred_1[2] + theta_4*data_pred_1[3] + theta_5*data_pred_1[4] + theta_6*data_pred_1[5] + theta_7*data_pred_1[6] + theta_8*data_pred_1[7] + theta_9*data_pred_1[8]))))
y_pred_2 = (1/(1 + math.exp(-(theta_0 + theta_1*data_pred_2[0] + theta_2*data_pred_2[1] + theta_3*data_pred_2[2] + theta_4*data_pred_2[3] + theta_5*data_pred_2[4] + theta_6*data_pred_2[5] + theta_7*data_pred_2[6] + theta_8*data_pred_2[7] + theta_9*data_pred_2[8]))))
```

```python
print ("Prediction for data_pred_1 = ", str(y_pred_1))
if y_pred_1 < 0.5:
  print("Prediction 1: benign")
else:
  print("Prediction 1: malignant")

print ("Prediction for data_pred_2 = ", str(y_pred_2))
if y_pred_2 < 0.5:
  print("Prediction 2: benign")
else:
  print("Prediction 2: malignant")
```

Run program & result:

```
diff_theta_0 = 0.021559449585472575, Theta 0 =  -2.4368114242284777

diff_theta_1 = -0.0019441663493601283, Theta 1 =  0.102986374108777

diff_theta_2 = -0.007818838372546776, Theta 2 =  1.4183973294679488

diff_theta_3 = 0.0005293235680550248, Theta 3 =  0.3679052350829014

diff_theta_4 = -0.0024458372823228137, Theta 4 =  0.3408869245069197

diff_theta_5 = 0.0011757342411032427, Theta 5 =  -0.4506476588444641

diff_theta_6 = -0.0033600971409567092, Theta 6 =  0.643928833584288

diff_theta_7 = 0.004468038018854651, Theta 7 =  -1.0703393838884085

diff_theta_8 = -0.007130564192882213, Theta 8 =  1.0750911154848202

diff_theta_9 = 0.013941573305666529, Theta 9 =  -1.5818424257029216

Prediction for data_pred_1 =  0.9999998077995643

Prediction 1: malignant

Prediction for data_pred_2 =  0.9999579826979751

Prediction 2: malignant
```

Therefore, the class of the two last records is malignant.

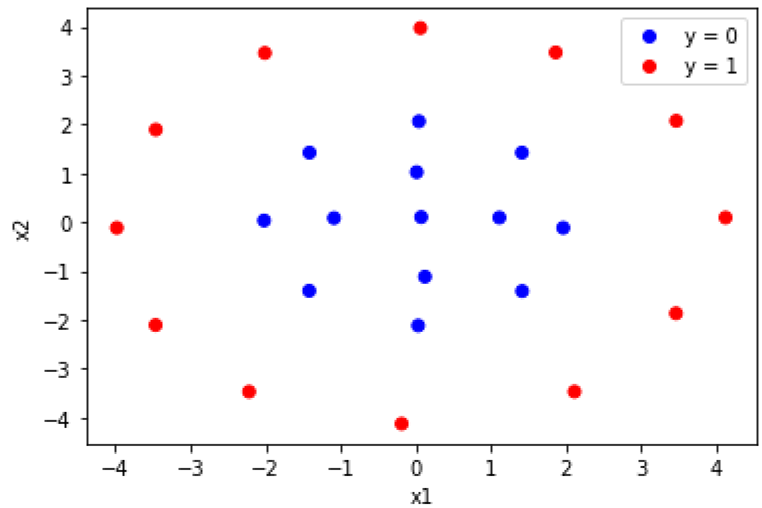
3.

Plot all points in two different classes:

Source code:

```
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import numpy as np
x1 = [-3.98, -3.464, -3.461, -2.22, -2.02, -2.01, -1.42, -1.416, -1.09, -
0.19, 0.01, 0.03, 0.04, 0.06, 0.07, 0.12, 1.11, 1.411, 1.414, 1.86, 1.96,
2.11, 3.461, 3.464, 4.12]
x2 = [-0.12, -2.11, 1.89, -3.474, 0.03, 3.459, -1.409, 1.419, 0.08, -4.13,
1.02, -2.12, 2.06, 3.97, 0.1, -1.12, 0.09, 1.419, -1.415, 3.47, -0.12, -
3.472, -1.87, 2.07, 0.09]
y = [1,1,1,1,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,1,0,1,1,1,1]

# if y = 0, the color is blue
# if y = 1, the color is red
# plot all x1, x2 points with color for y = 0 and y = 1
s  =  plt.scatter(x1,  x2,  c=y  ,cmap  =  mcolors.ListedColormap(["blue",
"red"]),marker='o')

# include legend for red and blue points
h, l = s.legend_elements()
plt.legend(h,("y = 0", "y = 1"))

plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```

Run program & result:



Based on the observations, we can see that the boundary decision function should be an eclipse function. We can see that the blue and red points create 2 eclipses in the graph.

We know that the eclipse function is:

$$\frac{x_1^{\ 2}}{a^2} + \frac{x_2^{\ 2}}{b^2} = 1$$

Therefore, we have a boundary decision function as below:

$$a^2b^2 - b^2x_1^{\ 2} - c^2x_2^{\ 2} = 0$$

Let $a^2 = \theta_1$ and $b^2 = \theta_2$, we have a hypothesis function as below:

$$h_0(x) = g(\theta_1\theta_2 - \theta_2x_1^{\ 2} - \theta_1x_2^{\ 2})$$

Where $g(z) = \dfrac{e^z}{1 + e^z}$

We have the cost function as below:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} * \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) * \log(h_\theta(x^{(i)}))]$$

And we also have the loss function:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) * x_j^{(i)^2}$$

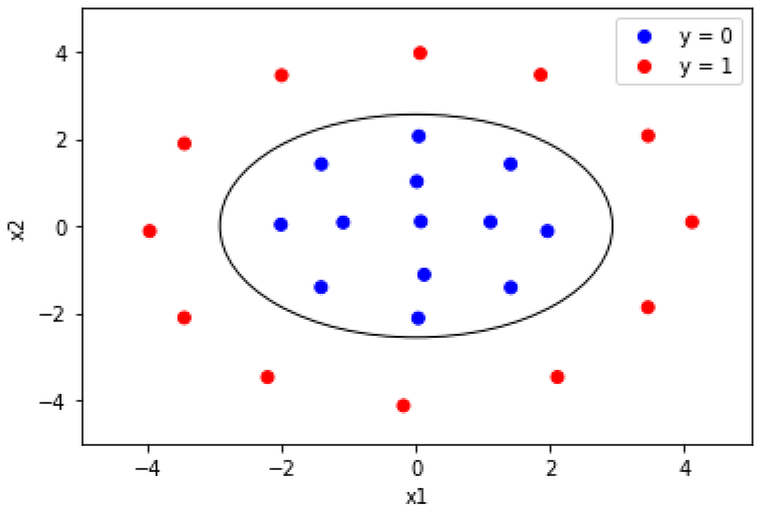
We have a source code below:

```
from matplotlib.patches import Ellipse
from numpy.ma.core import sqrt
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import math
import numpy as np
x1 = [-3.98, -3.464, -3.461, -2.22, -2.02, -2.01, -1.42, -1.416, -1.09, -
0.19, 0.01, 0.03, 0.04, 0.06, 0.07, 0.12, 1.11, 1.411, 1.414, 1.86, 1.96,
2.11, 3.461, 3.464, 4.12]
x2 = [-0.12, -2.11, 1.89, -3.474, 0.03, 3.459, -1.409, 1.419, 0.08, -4.13,
1.02, -2.12, 2.06, 3.97, 0.1, -1.12, 0.09, 1.419, -1.415, 3.47, -0.12, -
3.472, -1.87, 2.07, 0.09]
y = [1,1,1,1,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,1,0,1,1,1,1]

# if y = 0, the color is blue
# if y = 1, the color is red
# plot all x1, x2 points with color for y = 0 and y = 1
fig, ax = plt.subplots()
s  =  plt.scatter(x1,  x2,  c=y  ,cmap  =  mcolors.ListedColormap(["blue",
"red"]),marker='o')

# include legend for red and blue points
h, l = s.legend_elements()
plt.legend(h,("y = 0", "y = 1"))

plt.xlabel('x1')
plt.ylabel('x2')

theta_1 = theta_2 = 3
alpha = 0.0001
i = 0

while i <= 500000:
  diff_theta_1 = diff_theta_2 = 0
```

```
  for m in range (25):
    diff_theta_1    +=    (1/(1    +    math.exp(-(theta_1**2*theta_2**2    +
theta_2**2*(x1[m]**2) + theta_1**2*(x2[m]**2)))) - y[m]) * x2[m]
    diff_theta_2    +=    (1/(1    +    math.exp(-(theta_1**2*theta_2**2    +
theta_2**2*(x1[m]**2) + theta_1**2*(x2[m]**2)))) - y[m]) * x1[m]
  diff_theta_1 = diff_theta_1 * (1/25)
  diff_theta_2 = diff_theta_2 * (1/25)

  i += 1

  theta_1 = theta_1 - alpha * diff_theta_1
  theta_2 = theta_2 - alpha * diff_theta_2


print("diff_theta_1  =  "  +  str(diff_theta_1)  +  ",  "  +  "Theta  1  =  ",
str(theta_1))
print("diff_theta_2  =  "  +  str(diff_theta_2)  +  ",  "  +  "Theta  2  =  ",
str(theta_2))

ellipse = Ellipse((0,0), theta_1*2, theta_2*2, fill = False)
ax.add_artist(ellipse)

ax.set_xlim(-5,5)
ax.set_ylim(-5,5)

plt.show()
```

Run program & result:

```
diff_theta_1 = 0.0013600000000000057, Theta 1 =  2.9319998639247697

diff_theta_2 = 0.008760000000000039, Theta 2 =  2.561999123952988
```



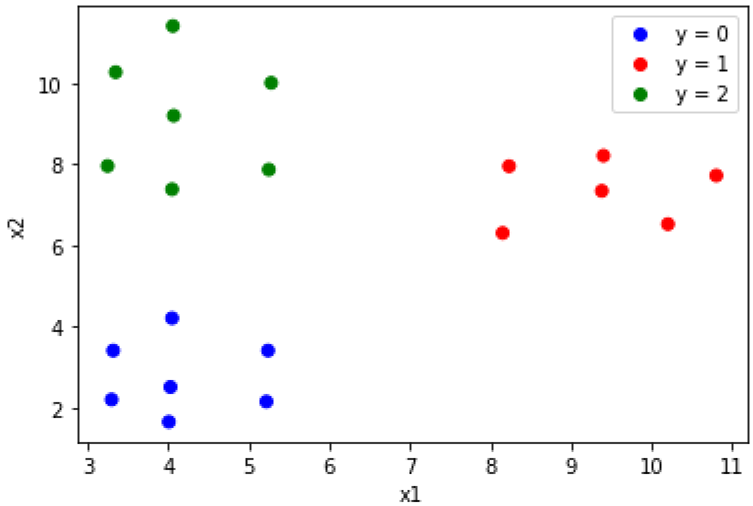
4.

Source code:

```
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import math
import numpy as np

x1 = [3.25, 3.3, 3.32, 3.35, 4.01, 4.03, 4.05, 4.05, 4.06, 4.07, 5.22, 5.24,
5.25, 5.28, 8.15, 8.23, 9.38, 9.4, 10.2, 10.8]
x2 = [7.956, 2.2, 3.41, 10.272, 1.65, 2.51, 4.21, 7.38, 11.412, 9.198, 2.15,
3.41, 7.866, 10.008, 6.3, 7.95, 7.34, 8.21, 6.52, 7.72]
y = [2, 0, 0, 2, 0, 0, 0, 2, 2, 2, 0, 0, 2, 2, 1, 1, 1, 1, 1, 1]

fig, ax = plt.subplots()
s = plt.scatter(x1, x2, c=y ,cmap = mcolors.ListedColormap(["blue", "red",
"green"]),marker='o')

# include legend for red and blue points
h, l = s.legend_elements()
plt.legend(h,("y = 0", "y = 1", "y = 2"))

plt.xlabel('x1')
```

```
plt.ylabel('x2')

plt.show()
```

Run program & we have the figure below:



Based on the observations, we have three hypothesis functions for the three classes.

$$h_\theta^{(1)} = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$$h_\theta^{(2)} = g(\theta_3 + \theta_4 x_1 + \theta_5 x_2)$$

$$h_\theta^{(3)} = g(\theta_6 + \theta_7 x_1 + \theta_8 x_2)$$

Where $g(z) = \dfrac{e^z}{1 + e^z}$

We have the three data tables for the classification of the three classes:

For the three classes, when y = 0, the value of the column y = 0 will be 1. If not, the value of the column y = 0 will be 0. The same classification is used for y = 1 and y = 2.

x1	x2	y	y = 0	y = 1	y = 2
3.25	7.956	2	0	0	1
3.3	2.2	0	1	0	0
3.32	3.41	0	1	0	0
3.35	10.272	2	0	0	1
4.01	1.65	0	1	0	0
4.03	2.51	0	1	0	0
4.05	4.21	0	1	0	0
4.05	7.38	2	0	0	1
4.06	11.412	2	0	0	1
4.07	9.198	2	0	0	1
5.22	2.15	0	1	0	0
5.24	3.41	0	1	0	0
5.25	7.866	2	0	0	1
5.28	10.008	2	0	0	1
8.15	6.3	1	0	1	0
8.23	7.95	1	0	1	0
9.38	7.34	1	0	1	0
9.4	8.21	1	0	1	0
10.2	6.52	1	0	1	0
10.8	7.72	1	0	1	0

We have the cost function formula as below:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} * \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) * \log(h_\theta(x^{(i)}))]$$

And we also have the loss function formula:

$$\frac{\partial}{\partial\theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) * x_j^{{(i)}^2}$$

Source code:

```
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import math
import numpy as np

x1 = [3.25, 3.3, 3.32, 3.35, 4.01, 4.03, 4.05, 4.05, 4.06, 4.07, 5.22, 5.24,
5.25, 5.28, 8.15, 8.23, 9.38, 9.4, 10.2, 10.8]
x2 = [7.956, 2.2, 3.41, 10.272, 1.65, 2.51, 4.21, 7.38, 11.412, 9.198, 2.15,
3.41, 7.866, 10.008, 6.3, 7.95, 7.34, 8.21, 6.52, 7.72]
y = [2, 0, 0, 2, 0, 0, 0, 2, 2, 2, 0, 0, 2, 2, 1, 1, 1, 1, 1, 1]
y0 = [0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
y1 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]
y2 = [1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]

theta_0 = theta_1 = theta_2 = theta_3 = theta_4 = theta_5 = theta_6 = theta_7
= theta_8 = 0
alpha = 0.0001
a = 0


while a < 36000:
  for i in range (0,20):
    diff_theta_0 = diff_theta_1 = diff_theta_2 = diff_theta_3 = diff_theta_4
= diff_theta_5 = diff_theta_6 = diff_theta_7 = diff_theta_8 = 0
    diff_theta_0  +=  (1/(1  +  math.exp(-(theta_0  +  theta_1*x1[i]  +
theta_2*x2[i]))) - y0[i])
    diff_theta_1  +=  (1/(1  +  math.exp(-(theta_0  +  theta_1*x1[i]  +
theta_2*x2[i]))) - y0[i]) * (x1[i])
    diff_theta_2  +=  (1/(1  +  math.exp(-(theta_0  +  theta_1*x1[i]  +
theta_2*x2[i]))) - y0[i]) * (x2[i])

    diff_theta_3  +=  (1/(1  +  math.exp(-(theta_3  +  theta_4*x1[i]  +
theta_5*x2[i])))  - y1[i])
    diff_theta_4  +=  (1/(1  +  math.exp(-(theta_3  +  theta_4*x1[i]  +
theta_5*x2[i]))) - y1[i]) * (x1[i])
    diff_theta_5  +=  (1/(1  +  math.exp(-(theta_3  +  theta_4*x1[i]  +
theta_5*x2[i]))) - y1[i]) * (x2[i])
```

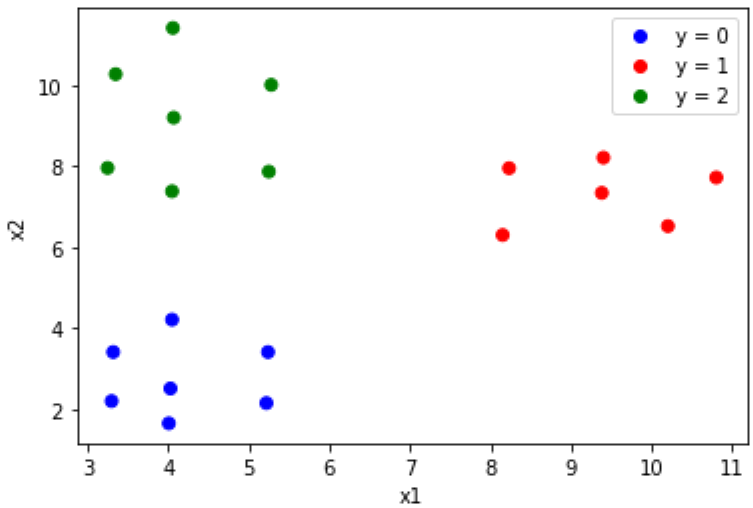
```
    diff_theta_6  +=  (1/(1  +  math.exp(-(theta_6  +  theta_7*x1[i]  +
theta_8*x2[i])))  - y2[i])
    diff_theta_7  +=  (1/(1  +  math.exp(-(theta_6  +  theta_7*x1[i]  +
theta_8*x2[i]))) - y2[i]) * (x1[i])
    diff_theta_8  +=  (1/(1  +  math.exp(-(theta_6  +  theta_7*x1[i]  +
theta_8*x2[i]))) - y2[i]) * (x2[i])

  diff_theta_0 = diff_theta_0 * (1/20)
  diff_theta_1 = diff_theta_1 * (1/20)
  diff_theta_2 = diff_theta_2 * (1/20)
  diff_theta_3 = diff_theta_3 * (1/20)
  diff_theta_4 = diff_theta_4 * (1/20)
  diff_theta_5 = diff_theta_5 * (1/20)
  diff_theta_6 = diff_theta_6 * (1/20)
  diff_theta_7 = diff_theta_7 * (1/20)
  diff_theta_8 = diff_theta_8 * (1/20)


  theta_0 = theta_0 - alpha * diff_theta_0
  theta_1 = theta_1 - alpha * diff_theta_1
  theta_2 = theta_2 - alpha * diff_theta_2
  theta_3 = theta_3 - alpha * diff_theta_3
  theta_4 = theta_4 - alpha * diff_theta_4
  theta_5 = theta_5 - alpha * diff_theta_5
  theta_6 = theta_6 - alpha * diff_theta_6
  theta_7 = theta_7 - alpha * diff_theta_7
  theta_8 = theta_8 - alpha * diff_theta_8
  a += 1

print("diff_theta_0  =  "  +  str(diff_theta_0)  +  ",  "  +  "Theta  0  =  ",
str(theta_0))
print("diff_theta_1  =  "  +  str(diff_theta_1)  +  ",  "  +  "Theta  1  =  ",
str(theta_1))
print("diff_theta_2  =  "  +  str(diff_theta_2)  +  ",  "  +  "Theta  2  =  ",
str(theta_2))
print("diff_theta_3  =  "  +  str(diff_theta_3)  +  ",  "  +  "Theta  3  =  ",
str(theta_3))
print("diff_theta_4  =  "  +  str(diff_theta_4)  +  ",  "  +  "Theta  4  =  ",
str(theta_4))
```

```
print("diff_theta_5 = " + str(diff_theta_5) + ", " + "Theta 5 = ",
str(theta_5))
print("diff_theta_6 = " + str(diff_theta_6) + ", " + "Theta 6 = ",
str(theta_6))
print("diff_theta_7 = " + str(diff_theta_7) + ", " + "Theta 7 = ",
str(theta_7))
print("diff_theta_8 = " + str(diff_theta_8) + ", " + "Theta 8 = ",
str(theta_8))

fig, ax = plt.subplots()
s = plt.scatter(x1, x2, c=y ,cmap = mcolors.ListedColormap(["blue", "red",
"green"]),marker='o')

# include legend for red and blue points
h, l = s.legend_elements()
plt.legend(h,("y = 0", "y = 1", "y = 2"))


plt.xlabel('x1')
plt.ylabel('x2')

plt.show()

x1_test = 4.01
x2_test = 3.02
test_y0 = (1/(1 + math.exp(-(theta_0 + theta_1*x1_test + theta_2*x2_test))))
test_y1 = (1/(1 + math.exp(-(theta_3 + theta_4*x1_test + theta_5*x2_test))))
test_y2 = (1/(1 + math.exp(-(theta_6 + theta_7*x1_test + theta_8*x2_test))))
print("Point1: (4.01, 3.02)")
print("Class 0: ", test_y0)
print("Class 1: ", test_y1)
print("Class 2: ", test_y2)

x1_test = 9.1
x2_test = 6.5
test_y0 = (1/(1 + math.exp(-(theta_0 + theta_1*x1_test + theta_2*x2_test))))
test_y1 = (1/(1 + math.exp(-(theta_3 + theta_4*x1_test + theta_5*x2_test))))
test_y2 = (1/(1 + math.exp(-(theta_6 + theta_7*x1_test + theta_8*x2_test))))
print("Point2: (9.1, 6.5)")
print("Class 0: ", test_y0)
```

```python
print("Class 1: ", test_y1)
print("Class 2: ", test_y2)

x1_test = 3.5
x2_test = 9.5
test_y0 = (1/(1 + math.exp(-(theta_0 + theta_1*x1_test + theta_2*x2_test))))
test_y1 = (1/(1 + math.exp(-(theta_3 + theta_4*x1_test + theta_5*x2_test))))
test_y2 = (1/(1 + math.exp(-(theta_6 + theta_7*x1_test + theta_8*x2_test))))
print("Point3: (3.50, 9.50)")
print("Class 0: ", test_y0)
print("Class 1: ", test_y1)
print("Class 2: ", test_y2)

x1_test = 6.01
x2_test = 6.01
test_y0 = (1/(1 + math.exp(-(theta_0 + theta_1*x1_test + theta_2*x2_test))))
test_y1 = (1/(1 + math.exp(-(theta_3 + theta_4*x1_test + theta_5*x2_test))))
test_y2 = (1/(1 + math.exp(-(theta_6 + theta_7*x1_test + theta_8*x2_test))))
print("Point4: (6.01, 6.01)")
print("Class 0: ", test_y0)
print("Class 1: ", test_y1)
print("Class 2: ", test_y2)
```

Run program & result:

```
diff_theta_0 = 0.0016383652464294189, Theta 0 =  -0.01909877838771963

diff_theta_1 = 0.017694344661437725, Theta 1 =  -0.20626680658737467

diff_theta_2 = 0.012648179702435114, Theta 2 =  -0.14744256915319728

diff_theta_3 = -0.001638365246429424, Theta 3 =  0.01909877838771963

diff_theta_4 = -0.01769434466143778, Theta 4 =  0.2062668065873747

diff_theta_5 = -0.012648179702435154, Theta 5 =  0.14744256915319728

diff_theta_6 = 0.0016383652464294189, Theta 6 =  -0.01909877838771963

diff_theta_7 = 0.017694344661437725, Theta 7 =  -0.20626680658737467

diff_theta_8 = 0.012648179702435114, Theta 8 =  -0.14744256915319728
```




```
Point1: (4.01, 3.02)

Class 0:  0.7844018562790571

Class 1:  0.2155981437209429

Class 2:  0.2155981437209429
```

Point1 belongs to class 0

```
Point2: (9.1, 6.5)

Class 0:  0.05444899612798037

Class 1:  0.9455510038720196

Class 2:  0.05444899612798037
```

Point2 belongs to class 1

```
Point3: (3.50, 9.50)

Class 0:  0.10510589675809119

Class 1:  0.10510589675809119

Class 2:  0.8948941032419089
```

Point3 belongs to class 2

```
Point4: (6.01, 6.01)
```

```
Class 0:  0.8951904998531087

Class 1:  0.10480950014689122

Class 2:  0.10480950014689122
```

Point4 belongs to class 0