

General Sir John Kotelawala Defense University

Department of Electrical, Electronics & Telecommunication Engineering

Machine Learning

ET 4103

Assignment – 02

Regularized Logistic Regression

Index No : D/ENG/22/0120/ET

Name : M. A. E. Wijesuriya

Intake : 39

Submission Date : 20/06/2025

**Q1. Utilize the given Jupyter notebook[1] for Regularized Logistic Regression. Comment on the code and the output of the program, explaining utilized Machine Learning concepts where necessary**

The following code is a python program that demonstrates Regularized Logistic Regression. Logistic Regression is a type of statistical model used to classify data into binary outcomes. It is a supervised learning algorithm that used a sigmoid function to generate probability values for a set of linear inputs. This probability value is then used to classify the data into one of two classes.

Regularized Logistic Regression utilizes a regularization parameter (lambda) that is used to prevent overfitting, by adding a penalty term to the cost function.

Code with Explanation:

(text in *italics,* along with any graphs or tables,are the output of the preceding code segment)

from google.colab import drive

drive.mount('/content/drive') # Grants Colab access to Google Drive in order to retrieve the data files

%cd "/content/drive/MyDrive/ML\_files"

*/content/drive/MyDrive/ML\_files*

# Importing Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# Retrieving the data file

data\_path = 'ex2data2.txt'

data = pd.read\_csv(data\_path, header=None, names = ["x1","x2","y"])

data.head() # Prints the first 5 rows of the data in a table

|  |  |  |  |
| --- | --- | --- | --- |
|  | x1 | x2 | y |
| 0 | 0.051267 | 0.69956 | 1 |
| 1 | -0.092742 | 0.68494 | 1 |
| 2 | -0.213710 | 0.69225 | 1 |
| 3 | -0.375000 | 0.50219 | 1 |
| 4 | -0.513250 | 0.46564 | 1 |

# Generates a scatter plot of the data with negative data marked with blue dots, and positive data marked with yellow crosses

def plotData(data, label\_x, label\_y, label\_pos, label\_neg, axes=None):

    # Get indexes for class 0 and class 1

    neg = data['y'] == 0

    pos = data['y'] == 1

    # If no specific axes object has been passed, get the current axes.

    if axes == None:

        axes = plt.gca()

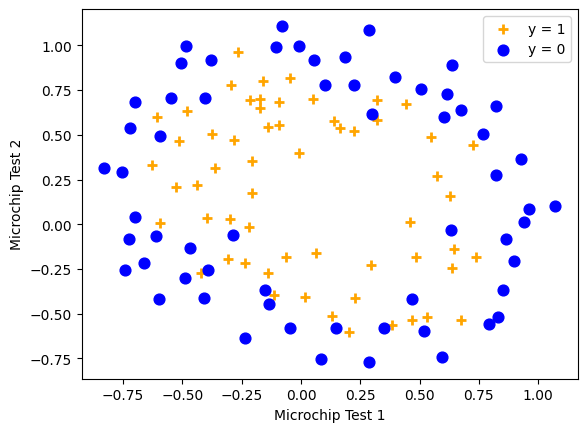
    axes.scatter(data[pos]['x1'], data[pos]['x2'], marker='+', c='orange', s=60, linewidth=2, label=label\_pos)

    axes.scatter(data[neg]['x1'], data[neg]['x2'], c='blue', s=60, label=label\_neg)

    axes.set\_xlabel(label\_x)

    axes.set\_ylabel(label\_y)

    axes.legend(frameon= True, fancybox = True);



n = data.shape[1]-1

x = data[data.columns[0:n]]

y = data[data.columns[n:n+1]]

# convert to np.array

X = x.values

y = y.values

## Feature mapping

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(6)

XX = poly.fit\_transform(X)

print(X.shape, XX.shape) # Shows the change in the array X before and after transformation

*(118, 2) (118, 28)*