**Faculty of Computing**



**Lab 04: Logistic Regression**

**CS471 Machine Learning**

**BESE – 13A**

**17th February 2025**

**Lab Engineer: Mr. Junaid Sajid**

## Instructor: Dr. Hashir Moheed Kiani

## Introduction

This laboratory exercise will focus on the python implementation of logistic regression. Logistic regression is a supervised learning technique that incorporates a sigmoid function activation with the linear regression algorithm to implement classification. Unlike regression, classification involves discrete labels such 0/1, true/false, cat/not a cat, benign/malignant etc. The sigmoid function causes the hypothesis values to take place between 0 or 1. Similar to regression, weight parameters are trained on a dataset so as to fit a model that can make accurate predictions from that dataset.

## Objectives

The following are the main objectives of this lab:

* Extract and prepare the training and validation datasets
* Use feature scaling to ensure uniformity among the feature columns
* Implement cost function to get the overall loss
* Implement gradient descent algorithm to update weight parameters
* Plot the training and validation losses
* Make scatter plots of the classification

**Theory**

Logistic Regression is another basic supervised learning technique besides Linear Regression. In logistic regression, the linear regression algorithm is modified by applying a sigmoid function to the predicted value. This causes the prediction to fall between 0 to 1 values. Thus, logistic regression is actually a classification technique built from the linear regression. The sigmoid function is a type of an activation function. Aside from loss and accuracy, logistic regression also at times, requires calculation of precision and recall. This is needed when the dataset is skewed; the two class labels are not equally distributed in the dataset.

**Lab Task 1 - Dataset Preparation**

Choose any dataset you want that feasible for logistic regression containing several feature columns. You will need to select any 3 of the feature columns to make your own dataset. Also, you will need to choose the label column that your model will predict. Ensure that the label column has binary values. Specify the features and label that you choose.

The dataset examples are to be divided into 2 separate portions: training and validation datasets (choose from 80-20 to 70-30 ratios). Save the prepared datasets as CSV files. Next, load the datasets into your python program and store them as NumPy arrays (Xtrain , ytrain, Xval, yval,).

**Lab Task 2 – Sigmoid Activation**

For logistic regression, you will implement the following hypothesis:

h(x) = g(w0 + w1x1 + w2x2 + w3x3 + …)

g(z) = 1 / (1 + e-z)

The w represents the weights and the x represent the features. h(x) is to be calculated for each training example and its difference with the label y of that training example will represent the loss. The g(z) function represents the sigmoid activation function. In this task, you will write a function that takes in a value z as argument and outputs the result of the sigmoid activation g(z). Provide the code for this task:

**Lab Task 3 - Cost Function**

In this task, you will write a cost function that calculates the overall loss across a set of training examples. This cost function will be useful to calculate the losses in both the training and validation phases of the program.

cost\_function(X, y)

The X and y are the features and labels of the training/validation dataset. The function will return the cost value. The cost function is given by:

The m is the number of the training/validation examples in the dataset. Remember that the hypothesis requires sigmoid activation. Write the code for the cost function and implement it to print out the cost. You will need to initialize the weights to some random values in order to calculate the hypothesis. Provide the code and all relevant screenshots showcasing the use of your cost function.

**Lab Task 4 – Gradient Descent**

In this task, you will write a function that uses gradient descent to update the weight parameters:

gradient\_descent(X, y, alpha)

The X and y are the features and labels of the training dataset, *alpha* is the learning rate which is a tuning hyperparameter. The gradient descent algorithm is given as follows:

The gradient descent for logistic regression may seem identical to that in linear regression, however, it should be noted that they are not the same formulas as the cost function for the logistic regression is different from that of linear regression.

For the submission, you will need to run the gradient descent algorithm once to update the weights. You will need to print the weights, training cost and validation cost both before and after the weight update. Provide the code and all relevant screenshots of the final output.

**Lab Task 5 – Training and Validation Implementation**

In this task, you will use the functions from the previous two tasks to write a “main” function that performs the actual training and validation. Use the cost function and gradient descent function on the training examples to determine the training loss and update the weights. Then, use the cost function on the validation examples to determine the validation loss. This single iteration over the entire dataset (both training and validation) marks completion of one epoch. You will need to perform the training and validation over several epochs (the epoch number is another hyperparameter that must be chosen). Ensure that at the end of each epoch, the training loss, validation loss, precision value and recall value are stored for plotting purposes. To get the precision and recall, use the following:

Start the training at some value of alpha. Try multiple training attempts with varying alpha values and find the best value of the alpha. Once you have found the best alpha value, showcase the output by making three plots:

1. training loss and validation loss vs. the epoch number
2. precision and recall vs. the epoch number
3. precision vs. recall

Ensure all axes are labeled appropriately. Provide the code (excluding function definitions) and all plots of the final output.

**Lab Task 6 – Prediction and Scatter Plot**

Save the weights that fit the best model and use them to create a prediction function. The prediction function will take the features as input and output the predicted class of the label. To convert the output to 0 or 1, a threshold of 0.5 needs to be applied. Call your prediction function by giving it some input values to make at least three predictions. Print your predictions and take a screenshot. Additionally, your program must make a scatter plot showing the training and validation examples. The coordinates in the scatter plot correspond to the inputs (x). The class is denoted by (red) o for zero and (blue) x for one values. The predicted value must be shown as a black O or X. Provide the code and screenshot of it being used.