**Faculty of Computing**



**Lab 03: Linear Regression**

**CS471 Machine Learning**

**BESE – 13A**

**10th February 2025**

**Lab Engineer: Mr. Junaid Sajid**

## Instructor: Dr. Hashir Moheed Kiani

## Introduction

This laboratory exercises the python implementation of linear regression. Linear regression is a basic supervised learning technique in which parameters are trained on a dataset to fit a model that best approximates that dataset.

## Objectives

The following are the main objectives of this lab:

* Extract and prepare the training and cross-validation datasets
* Use feature scaling to ensure uniformity among the feature columns
* Implement cost function on both training and cross-validation datasets
* Implement gradient descent algorithm
* Plot the training and cross-validation losses
* Use L2 regularization to counter overfitting

**Theory**

Linear Regression is a very basic supervised learning technique. To calculate the loss in each training example, the difference between a hypothesis and the label (y) is calculated. The hypothesis is a linear equation of the features (x) in the dataset with the coefficients acting as the weight parameters. These weight parameters are initialized to random values at the start but are then trained over time to learn the model. The cost function is used to calculated the error between the predicted y^ and the actual y.

A major problem in the training is that the weights that are trained may fit the model for only the data it is given. This means that the model will not generalize to examples outside the dataset and is referred to as “overfitting”. Such overfitting makes the machine learning implementation very impractical for real-life applications where data has high variation. To prevent overfitting of the model, a modification in the cost function and gradient descent is implemented. This modification is called regularization and is itself controlled by a hyperparameter (lambda).

**Lab Task 1 - Dataset Preparation, Feature Scaling**

You have been provided with a dataset (California Housing Dataset) containing several feature columns. You will need to select any 3 of the feature columns to make your own dataset. The “MedHouseVal” is the label column that your model will predict. The dataset examples are to be divided into 2 separate portions: training and cross-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,). Next, use feature scaling to rescale the feature columns of both datasets so that their values range from 0 to 1. Finally, print both of the datasets (you need to show any 5 rows of the datasets).

**Lab Task 2 - Cost Function without and with Regularization**

For linear regression, you will implement the following hypothesis:

h(x) = w0 + w1x1 + w2x2 + w3x3 + …

The wj represent the weights while the xj represents the jth feature. The linear hypothesis 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. In this task, you will write a cost function that calculates the overall loss across a set of examples. This cost function will be useful to calculate the losses in both the training and cross-validation phases of the program.

cost\_function(X, y, lambd)

The X and y are the features and labels of either the training or the cross-validation datasets. This is useful as it can be used for either the training examples or the cross-validation examples of the dataset. The *lambd* is the regularization parameter (Note that *lambda* is a keyword reserved in python). The function will calculate the losses to return the overall cost value. The cost function is given by:

The m is the number of the examples in the dataset and n is the total number of features (or non-bias weights) in the hypothesis. Write the code for the cost function and implement it for your training and cross-validation datasets to print out the cost. Provide the code and all relevant screenshots of the final output.

**Lab Task 3 –Gradient Descent without and with Regularization**

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

gradient\_descent(X, y, alpha, lambd)

The *alpha* is the learning rate (hyperparameter 1) and *lambd* is the regularization parameter (hyperparameter 2). The gradient descent algorithm is given as follows:

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 4 – Training and Validation Program**

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 respectively. Then, use the cost function on the cross-validation examples to determine the cross-validation loss. This single iteration over the entire dataset (both training and cross-validation) marks the completion of one epoch. You will need to perform the training and cross-validation over several epochs (the epoch number is another hyperparameter that must be chosen). Ensure that at the end of each epoch, the training and cross-validation losses are stored for plotting purposes. When the final epoch is performed, note down the trained parameters (weights and bias) and make plot of the training and cross-validation losses (y-axis) over the epochs (x-axis). Ensure that both of the losses appear on the same graph. You only need to show a single plot for this task. Provide the code (excluding function definitions) and all relevant screenshots of the final output.

**Lab Task 5 – Tuning Alpha and Lambda**

In this task, you will use your linear regression code from the previous task. Tune the alpha and lambda hyperparameters at different values to get several plots. You need to get at least 6 plots. Mention the alpha and lambda values in the plot titles. Ensure all axes are labeled appropriately.