**Department of Electrical Engineering**

**Faculty Member:** LE Munadi Sial **Date:** 25-Oct-2023

**Semester:** 7th **Group:**

# CS471 Machine Learning

**Lab 7: Linear Regression II – Train-Test Split and Regularization**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO4 - CLO4** | **PLO4 -CLO4** | **PLO5 -CLO5** | **PLO8 -CLO6** | **PLO9 -CLO7** |
| **Name** | **Reg. No** | **Viva /Quiz / Lab Performance** | **Analysis of data in Lab Report** | **Modern Tool Usage** | **Ethics** | **Individual and Team Work** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

## Introduction

This laboratory exercise will extend the python implementation of linear regression performed in the previous lab. 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. The problem with using simple linear regression is that the trained models can overfit the dataset at which point regularization must be used to prevent overfitting. This lab will focus on integrating the regularization concept into the gradient descent algorithm.

## Objectives

The following are the main objectives of this lab:

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

## Lab Conduct

* Respect faculty and peers through speech and actions
* The lab faculty will be available to assist the students. In case some aspect of the lab experiment is not understood, the students are advised to seek help from the faculty.
* In the tasks, there are commented lines such as #YOUR CODE STARTS HERE# where you have to provide the code. You must put the code/screenshot/plot between the #START and #END parts of these commented lines. Do NOT remove the commented lines.
* Use the tab key to provide the indentation in python.
* When you provide the code in the report, keep the font size at 12

**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).

A brief summary of the relevant keywords and functions in python is provided below:

**print()** output text on console

**input()** get input from user on console

**range()**  create a sequence of numbers

**len()** gives the number of characters in a string

**if** contains code that executes depending on a logical condition

**else** connects with **if** and **elif**, executes when conditions are not met

**elif** equivalent to **else if**

**while** loops code as long as a condition is true

**for** loops code through a sequence of items in an iterable object

**break** exit loop immediately

**continue** jump to the next iteration of the loop

**def** used to define a function

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

You have been provided with a dataset containing several feature columns. You will need to select any 3 of the feature columns to make your own dataset. The “Sale Price” is the label column that your model will predict. The dataset examples are to be divided into 2 separate portions: training and test 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, Xtest, ytest,). 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).

***### TASK 1 CODE STARTS HERE ###***

*### TASK 1 CODE ENDS HERE ###*

***### TASK 1 SCREENSHOT STARTS HERE ###***

*### TASK 1 SCREENSHOT ENDS HERE ###*

**Lab Task 2 - Cost Function with Regularization \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

For linear regression, you will implement the following hypothesis:

h(x) = w0 + w1x1 + w2x2 + w3x3 + …

The wj and b 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 test phases of the program.

cost\_function(X, y, lambd)

The X and y are the features and labels of either the training or the test datasets. This is useful as it can be used for either the training examples or the test 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 test datasets to print out the cost. Provide the code and all relevant screenshots of the final output.

***### TASK 2 CODE STARTS HERE ###***

*### TASK 2 CODE ENDS HERE ###*

***### TASK 2 SCREENSHOT STARTS HERE ###***

*### TASK 2 SCREENSHOT ENDS HERE ###*

**Lab Task 3 –Gradient Descent 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:

Provide the code and all relevant screenshots of the final output.

***### TASK 3 CODE STARTS HERE ###***

*### TASK 3 CODE ENDS HERE ###*

***### TASK 3 SCREENSHOT STARTS HERE ###***

*### TASK 3 SCREENSHOT ENDS HERE ###*

**Lab Task 4 – Training and Testing 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 testing. 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 test examples to determine the test loss. This single iteration over the entire dataset (both training and test) marks completion of one epoch. You will need to perform the training and testing over several epochs (the epoch number is another hyperparameter that must be chosen). Ensure that at the end of each epoch, the training and test losses are stored for plotting purposes. When the final epoch is performed, note down the trained parameters (weights and bias) and makes plot of the training and test 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.

***### TASK 4 CODE STARTS HERE ###***

*### TASK 4 CODE ENDS HERE ###*

***### TASK 4 SCREENSHOT STARTS HERE ###***

*### TASK 4 SCREENSHOT ENDS HERE ###*

**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.

***### TASK 5 PLOTS START HERE ###***

*### TASK 5 PLOTS END HERE ###*