LAHORE UNIVERSITY OF MANAGEMENT SCIENCES
Syed Babar Ali School of Science and Engineering

EE212 Mathematical Foundations for Machine Learning and Data Science
Summer Semester 2020

Laboratory 3 – Least Square Applications

Issued: Wednesday 22 July, 2020

Total Marks: 50

Contribution to Final Assessment: 1.5%

Submission: Wednesday 22 July, 2020.

Goal

The goal of this laboratory is to find least squares solutions and apply data fitting in real life problems.

Instructions

If you have any concerns, you can ask us in the live zoom session, or in the chat. Each of you has been allotted TA/RA, so when you are done with the lab, let them know and they will mark it. It is your responsibility to ensure you get your work checked.

Name your files Task1.py, Task2.py and so on. Compress them in a **single** file and name it as LabXX_YourRollNumber. Submit this file on LMS before the deadline. No late submissions will be accepted.

Before starting, import the following libraries from python:

```
import numpy as np
import scipy
from matplotlib.image import imread
import pandas as pd
import matplotlib.pyplot as plt
import math
```

Task 1: Least Squares Applications (35 marks)

This lab task consists of two parts:

- In the first part, you will simply apply least squares approximation for different models on the noisy data set provided to you. You will learn the parameters using training data and fit your model on testing data.
- In the second part, you will see which approach to use for calculating the least squares solution, given the dimensions and rank of the matrix of your system of

linear equations.

Least squares approximation is a method for estimating the value of some parameters from the given noisy data. It can be seen as estimating that line which minimizes the sum of the squared distances (deviations) from the line of each observation. That is, for the relation $\boldsymbol{Ax} = \boldsymbol{y}$, we wish to find an $\boldsymbol{x_{ls}}$ such that we minimize the ℓ_2 - norm squared error. This is mathematically represented as:

$$\text{minimize} \quad \|\boldsymbol{Ax_{ls}} - \boldsymbol{y}\|_2^2$$

Model Under Consideration

We will be focusing on a very specific model called *Linear in Parameter (LIP)* polynomial model. This simply means that the model equation that relates the input to the output is of the form

$$f(t_i) = x_1 + x_2 t_i + x_3 t_i^2 + ... + x_M t_i^{M-1},$$

where t_i represents the data-point (input), $f(t_i)$ is the observation (output), $x_1, x_2, \ldots, x_M$ are the parameters of the model we wish to estimate and M is the order of polynomial. We note here that the output is non-linearly related to the input. However, the model is linear in terms of model parameters.

We assume that we have N outputs that we stack in a vector $\boldsymbol{y} = [f(t_1), f(t_2), ..., f(t_N)] \in \mathbf{R}^N$ which can be expressed in the matrix form using the model equation as

$$\boldsymbol{y} = \boldsymbol{Ax},$$

where $\boldsymbol{x} = [x_1, x_2, ..., x_M]^T \in \mathbf{R}^M$ and the matrix $\boldsymbol{A} \in \mathbf{R}^{N \times M}$ is given by

$$\boldsymbol{A} = \begin{bmatrix} 1 & t_1 & t_1^2 & \ldots t_1^M \\ 1 & t_2 & t_2^2 & \ldots t_2^M \\ \vdots & \vdots & \vdots & \vdots\vdots \\ 1 & t_N & t_N^2 & \ldots t_N^M \end{bmatrix}.$$

Problem Formulation

Given N data points (inputs) and corresponding N outputs related through a measurement model given above, we consider a problem to determine $\boldsymbol{x_{ls}}$ such that the error between $\boldsymbol{Ax_{ls}}$ and $\boldsymbol{y}$ is minimized in least-squares sense. Mathematically, we express this as

$$\text{minimize} \quad \|\boldsymbol{Ax_{ls}} - \boldsymbol{y}\|_2^2.$$

Implementation

For this task, we will be using a present-day scenario in order to make it more engaging. We will be analysing the growth in the number of COVID-19 cases over time, and try to (naively) predict the trends based on the current data we have. Here, $\boldsymbol{t}$ represents days since the outbreak and $\boldsymbol{y}$ represents the number of active cases in Pakistan (both of which have been compressed). The matrix $\boldsymbol{A}$ represents a model which we will keep changing in the following parts in order to find out which model fits the data best and which ones lead to overfitting.

1. Download the datasets ‘Training.csv’ and ‘Testing.csv’ from LMS. It consists of several data points in (t,y) form. To load this data set into numpy arrays, use the following commands (only load training data for now):

```
data = pd.read_csv("C:/Users/....../Training.csv")
data.head()
t = data.t.values
y = data.y.values
```

2. Lets begin with a simple linear model $y_i = x_1 + x_2 t_i$. Construct $\boldsymbol{A}$ for this linear model. Plot the data points and resulting equation on the same graph using the following commands:

```
fig, ax = plt.subplots(1,1, figsize=(10,6))
ax.plot(t, y, ‘.’, alpha=0.8, label=‘Data Points’)
ax.plot(t, y_ls, lw=1, label=‘Least Squares Eq’)
ax.legend(loc=‘upper left’)

ax.set_xlabel(‘$t$’)
ax.set_ylabel(‘$y$’)

fig.tight_layout()
plt.show()
```

   Here, $\boldsymbol{y_{ls}} = \boldsymbol{A}\boldsymbol{x_{ls}}$.

3. To quantitatively judge how accurate the data fitting is, calculate the error:

$$\epsilon = \|\boldsymbol{y} - \boldsymbol{y_{ls}}\|_2^2$$

4. Now to make this code more versatile, lets alter it so it can work for a model of any order of polynomial. Create a variable *poly*, which will be the degree of the polynomial used in your model, and modify the rest of the code to construct $\boldsymbol{A}$ and perform data fitting (and plotting) in accordance with its value. *(Hint: Use for loops)*

5. Plot the response for the following polynomial degrees:

   - $poly = 1$
   - $poly = 4$
   - $poly = 9$
   - $poly = 10$

   Also note the error for each *poly* and plot a graph of ϵ vs *poly*.

Note that technically the error might decreases with higher degree models, but it introduces the issue of overfitting. Overfitting is when we use a model more complex than the required model so that we could train it for every small detail in the training data, however when it is presented with a data different from the training data, the model is unable to give accurate predictions.

To see which of the above models is accurate and which ones cause overfitting, we will test them on our testing data. Load the testing data using:

```
data_test = pd.read_csv("C:/Users/....../Testing.csv")
data_test.head()
t_test = data_test.t.values
y_test = data_test.y.values
```

For each *poly*, you would have the corresponding $\boldsymbol{x_{ls}}$ that you learnt in the previous part using the training data. Construct a new $\boldsymbol{A}$ using $\boldsymbol{t}$ from the testing data. Calculate $\boldsymbol{y_{ls}}$ as:

$$\boldsymbol{y_{ls}} = \boldsymbol{A}\boldsymbol{x_{ls}}$$

This is your models' prediction. Now plot it along with the testing data to see how well your predictions fits the actual results:

```
fig, ax = plt.subplots(1,1, figsize=(10,6))
ax.plot(t_test, y_test, ‘.’, alpha=0.8, label=‘Data Points’)
ax.plot(t_test, y_ls, lw=1, label=‘Prediction’)
ax.legend(loc=‘upper left’)

ax.set_xlabel(‘$t$’)
ax.set_ylabel(‘$y$’)

fig.tight_layout()
plt.show()
```

Also calculate the error:

$$\epsilon = \|\boldsymbol{y} - \boldsymbol{y_{ls}}\|_2^2$$

Do this process for each *poly* you used for modeling. Plot the graph of ϵ vs *poly* and then decide which model gives the best result and which model causes overfitting.

Task 2: Regularized Least Squares (15 marks)

This task is meant to introduce ways to solve the equation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, based on what $\boldsymbol{A}$ is. We start of with the simple scenario where $\boldsymbol{A}$ is a square matrix.

1. Find the solution to the given system of linear equations:

$$2x + y - 2z = 3$$

$$x - y - z = 0$$

$$x + y + 3z = 12$$

Hint: Represent the system as $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, you'll see $\boldsymbol{A}$ is a square matrix so conventional method for taking inverse will suffice.

2. What if we had an overdetermined system that has no solution? We would need the least square solution. Find the least squares solution of the following system of linear equations:

$$2x = 1$$

$$-x + y = 0$$

$$2y = -1$$

Note: The least square solution will only be a unique solution if the columns of matrix $\boldsymbol{A}$ *are* ***linearly independent****, which they are in this case.*

3. What if the columns of $\boldsymbol{A}$ are not linearly independent? Well then the matrix $\boldsymbol{A}$ is not full-rank and is said to be *ill conditioned.* To get a unique least squares solution now, we use the following equation:

$$(\boldsymbol{A^T A} + \lambda \boldsymbol{I})\boldsymbol{x} = \boldsymbol{A^T y}$$

Here λ is the tuning parameter and its value is kept very small, as larger λ gives us an increased error. $\boldsymbol{I}$ is an identity matrix with the same dimensions as that of $\boldsymbol{A^T A}$. This method gives us the **Regularized Least Squares Solution**, which is a unique least squares solution for a particular λ. Notice how $\lambda = 0$ will give us the conventional least squares solution.

Now, for the following system of equations, check if $\boldsymbol{A}$ is full-rank or not. If it isn't then find the regularized least squares solution:

$$2x - 2y = 1$$

$$-x + y = 0$$

$$-2x + 2y = -1$$

Calculate the corresponding error, for different values of λ:

$$\epsilon = ||\boldsymbol{y} - \boldsymbol{y_{ls}}||_2^2$$

Plot ϵ vs λ.