Subject : EIE2108 Lab 2 Report

Student ID: 19069748D

Name: Kwok Kevin

Background of Lab 2:

Linear prediction is a mathematical operation where future values of a discrete-time signal are estimated as a linear function of previous samples. In digital signal processing, linear prediction is often called linear predictive coding (LPC). One of its applications is speech coding.

When we scan a sequence of values, we can predict the value to be observed with the previously observed values as follows:

$$\hat{x}(n) = \sum_{i=1}^{p} a_i x(n-i)$$

where $\hat{x}(n)$ is the predicted value of $x(n)$, $x(n)$ is the current value (i.e. the value to be observed), $x(n-i)$ is the i^{th} previously observed value, a_i is a fixed weighting coefficient, and p is the number of previous values used in the prediction.

The total square prediction error of the whole sequence is defined as

$$J = \sum_{n=1}^{N} (x(n) - \hat{x}(n))^2 = \sum_{n=1}^{N} (x(n) - \sum_{i=1}^{p} a_i x(n-i))^2$$

where N is he total number of values in the sequence. We would like to find the best set of a_i for $i = 1,2 \ldots p$ to minimize the total square prediction error J.

Mission of lab 2:

- With the case: $\hat{x}(n) = a_2 x(n-2) + a_3 x(n-3)$
- Derive gradients $\frac{\partial J}{\partial a_i}$ for i=2,3
- Take a python program to find out the best a_i for i=2,3 by making use of the gradient descent method.
- Plot a graph which shows the estimation of a_i's and the prediction error changes with iterations.

Analysis and implementation of Codes:

```
import numpy as np
import matplotlib.pyplot as plt
```

First of all, import the needed library for later usage

NumPy for the calculus part

Matplotlib for the plotting graph

```
def gradient(textdata):

    #make the dataset to suit with the formulas
    #take the first 3 number
    dataset_1 = np.delete(textdata, [0,1,2])

    #take the first and the last 2 number
    dataset_2 = np.delete(textdata, [0, len(textdata)-1, len(textdata)-2])

    #take the last 3 number
    dataset_3 = np.delete(textdata, [len(textdata)-1, len(textdata)-2, len(textdata)-3])



    #setting up variables
    a2 = 0
    a3 = 0

    #max number of interations
    max_no_of_iter = 50
    aa = 0.05

    #Take out the first 3 number and align the dataset
    x = len(textdata) - 3


    #data list
    costlist = []
    iterationslist = []



    for i in range(max_no_of_iter):

        #Gradient Descent Methods
        predict = a2 * dataset_2 + a3 * dataset_3

        #dy/dk1
        dyda2 =-(2/x)*sum(dataset_2 *(dataset_1 -predict))

        #dy/dk2
        dyda3 =-(2/x)*sum(dataset_3 *(dataset_1 -predict))

        a2 -= aa * dyda2
        a3 -= aa * dyda3
        cost = sum([error ** 2 for error in (dataset_1 - predict)])

        #Save the results in the list for plotting graph
        costlist.append(cost)
        iterationslist.append(i)


    #Plot out the graph
    fig, ax = plt.subplots()
    ax.plot(iterationslist, costlist, color = 'magenta', linewidth = 5, label = 'cost J')
    ax.set_title('Linear prediction error')
    ax.set_xlabel('iter')
    ax.set_ylabel('cost J')
    ax.grid(axis = 'x')
    ax.ticklabel_format(axis="y", style="scientific", scilimits=(2,2), useMathText = True)
    ax.legend()
    fig

    #check the best of  a2,a3, cost
    print("a2 is {}\na3 is {}".format(a2,a3))
    print("cost is {}".format(cost))
```

Second, defining a function for taking the linear prediction. And also develop a graph plotting inside the function. These two parts mainly in this customize function.
(cut out of the screen from the same photo for indicate the step and the codes)

```
#make the dataset to suit with the formulas
#take the first 3 number
dataset_1 = np.delete(textdata, [0,1,2])

#take the first and the last 2 number
dataset_2 = np.delete(textdata, [0, len(textdata)-1, len(textdata)-2])

#take the last 3 number
dataset_3 = np.delete(textdata, [len(textdata)-1, len(textdata)-2, len(textdata)-3])



#setting up variables
a2 = 0
a3 = 0

#max number of interations
max_no_of_iter = 50
aa = 0.05

#Take out the first 3 number and align the dataset
x = len(textdata) - 3


#data list
costlist = []
iterationslist = []
```

1. Set up 3 dataset which are taking from the text data file.
2. Set up a2 and a3 variables as same as the variables a2 a3 of

   $\hat{x}(n) = a_2 x(n-2) + a_3 x(n-3)$ this formula.
3. Set up the maximum number of interations
4. Take out the first 3 number and align the dataset.
5. List the data

Cont'd for function gradient:

(cut out of the screen from the same photo for indicate the step and the codes)

```
for i in range(max_no_of_iter):

    #Gradient Descent Methods
    predict = a2 * dataset_2 + a3 * dataset_3

    #dy/dk1
    dyda2 =-(2/x)*sum(dataset_2 *(dataset_1 -predict))

    #dy/dk2
    dyda3 =-(2/x)*sum(dataset_3 *(dataset_1 -predict))

    a2 -= aa * dyda2
    a3 -= aa * dyda3
    cost = sum([error ** 2 for error in (dataset_1 - predict)])

    #Save the results in the list for plotting graph
    costlist.append(cost)
    iterationslist.append(i)
```

This part is taking the for loop to complete the dy/dx for a2 and a3 by using gradient descent methods and store the results for plotting graph later.

(cut out of the screen from the same photo for indicate the step and the codes)

```
#Plot out the graph
fig, ax = plt.subplots()
ax.plot(iterationslist, costlist, color = 'magenta', linewidth = 5, label = 'cost J')
ax.set_title('linear prediction error')
ax.set_xlabel('iter')
ax.set_ylabel('cost J')
ax.grid(axis = 'x')
ax.ticklabel_format(axis="y", style="scientific", scilimits=(2,2), useMathText = True)
ax.legend()
fig
```

And then make the plotting setting for plotting the the result from the text data and gradient descent methods.

(cut out of the screen from the same photo for indicate the step and the codes)

```
#check the best of  a2,a3, cost
print("a2 is {}\na3 is {}".format(a2,a3))
print("cost is {}".format(cost))
```

Print the values of a2,a3 and the cost

Last part of the program

```
#take the data from the txt(1-5) file
data = np.array([])

with open('eie2108-Lab-2021-datafile-03.txt', 'r') as txt:
    for line in txt.readlines():
        data = np.append(data, float(line.strip()))

#run the Gradient Descent Method
gradient(data)
```

Take the data from the data files

Open the required file(1-5) select one for this case is running datafile-03

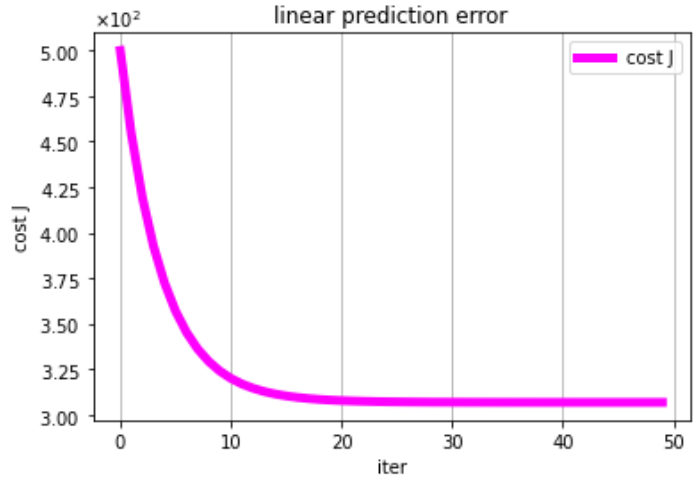
Since I am lazy to find a way do 5 job in once time so I decide to do it one by one.

And then the function that I built up before

Then there are outputs of the the program with 5 datafiles.

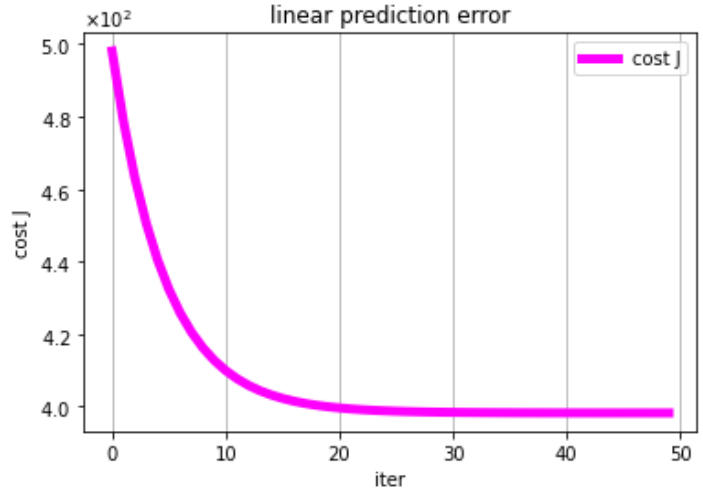
Output of lab 2 code (datafile-01-5):

For Data set of “ eie2108-lab-2021-datafile-01”:



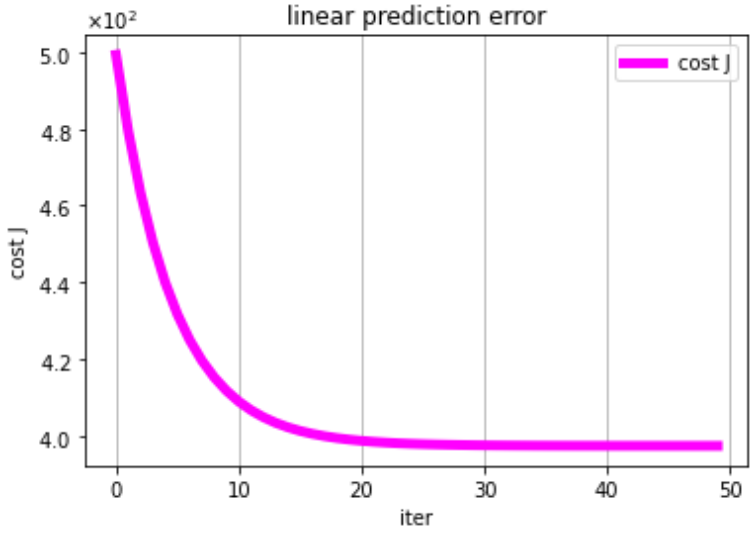

```
a2 is 0.45486839609378116
a3 is -0.3177044849935706
cost is 306.67375106693487
```

For Data set of ” eie2108-lab-2021-datafile-02”:



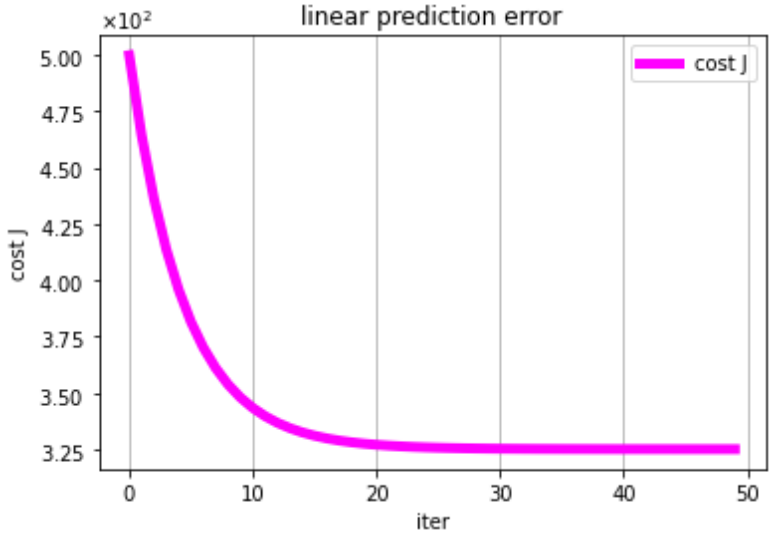

```
a2 is 0.4375391280035311
a3 is -0.06264733344989126
cost is 398.2306015528978
```

For Data set of ” eie2108-lab-2021-datafile-03”:



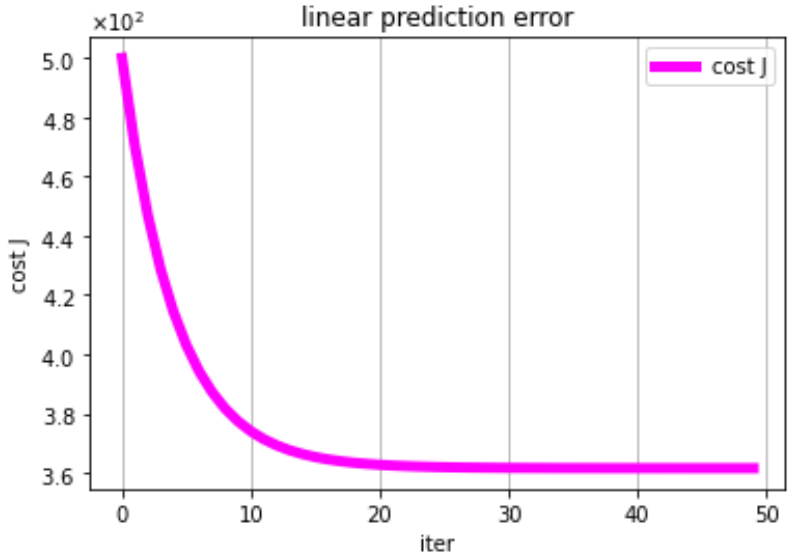

```
a2 is -0.3723837371937056
a3 is 0.24085197361049732
cost is 397.4098330314989
```

For Data set of ” eie2108-lab-2021-datafile-04”:




```
k1 is -0.5146082258672278
k2 is 0.24789813714949263
cost is 325.115504346818
```

For Data set of ” eie2108-lab-2021-datafile-05”:




```
a2 is 0.28878227005533996
a3 is -0.4003965245754031
cost is 361.54360979126255
```