Project 2 – Linear Regression

Rick Alayza, Trincy Thomas Kozhikkadan, Krystofer Newman

***Abstract*—There are several ways to generate a model for the specific dataset. Linear regression is the process of generating that model. This report used two methods to compare the correlation coefficients. Least squares approximation and Adam Optimization were the methods used to extract a model. The least squares approximation was slightly better with R = 0.0327 compared R = 0.0321 for the Adam Optimization.**

***Index Terms*—Adam Optimization, Gradient Decent, Least Squares Approximation**

I. Introduction

LINEAR Regression is the process of modeling the dependent variables to the independent variables, similar to mapping. This technique is commonly applied in computer learning. For this example, models were generated using two different methods for a dataset of 275 points. The features associated with the input were irrelevant. The intended purpose was to select the model that had the greater chance of modeling the next 30 points with the minimum amount of error. Least squares approximation and Adam Optimization were the two methods used to model the dataset. This report will have an overview of the mathematical theory, how each model was calculated, and the results. A correlation coefficient (R^2) will be calculated for each linear fit for comparison. The model with the best value, or fit, will then be formally submitted as the solution to the given problem.

II. Least Squares Approximation

The least square approximation is a linear algebra calculation to determine an independent value or vector that will minimize the error [1] [2]. The error, in this case, is the difference between the fitted model to the dataset values. This process used the matrix representation of the input features and the output. The first section will discuss the theory of least squares approximation followed by the results of using this method on the provided dataset.

A. Theory

The least square approximation implements the use of projections onto the column space. It is important to note the notation change for the purpose of this report. Equation 1 is the traditional notation commonly used in textbooks covering linear algebra. Equation 2 is the updated notation to reflect the use of the cost function, the predicted *x*, and predicted *y*.

Equation 3 is the Euclidean norm or distance notation. This projection of $\hat{y}$ onto Θ was equivalent to the minimized norm value as listed in Equation 4. Equation 5 is the fundamental equation for the system that minimizes the solution for $\hat{x}$.

$$Ax = b \tag{1}$$

$$\Theta\hat{x} = \hat{y} \tag{2}$$

$$\|\hat{y} - \Theta\hat{x}\| \tag{3}$$

$$\min_x \|\hat{y} - \Theta\hat{x}\| = P_{N(\Theta^H)}\hat{y} \tag{4}$$

$$\Theta^T\Theta\hat{x} = \Theta^T\hat{y} \tag{5}$$

B. Setup/Results

The given data was a singular vector of values and was treated as the output (Equation 6). In order to properly apply the least squares approximation, the matrix Θ had to be constructed. A vector of 1's was needed in the first column, also referred to as vector Θ_0. To complete the matrix, a second column was added, Θ_2, that was the index values for the output. Those index values were treated as the inputs to the singular feature of the system of equations (Equation 6). The next step was to manipulate the fundamental equation to solve for the minimal value. Equation 7 was the result of the equation manipulation. The code used to determine the model and plot the results is listed in the Appendix.

Figure 1 is the result of executing the least squares approximation code (Appendix). From the plot, it can be observed that there are several outliers, but the majority of the values are at and below 60.

$$\Theta = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ . & . \\ 1 & 275 \end{bmatrix} \qquad \hat{y} = \begin{bmatrix} 5 \\ 11 \\ . \\ 34.5 \end{bmatrix} \tag{6}$$

$$\hat{x} = (\Theta^T \cdot \Theta)^{-1} \cdot \Theta^{\mathrm{T}} \cdot \hat{y} \tag{7}$$

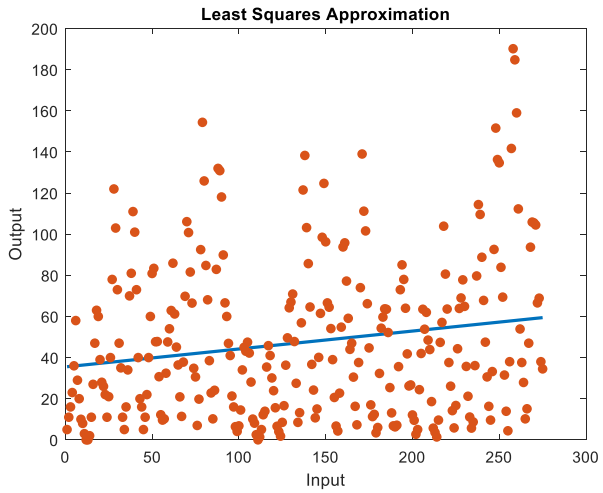

Figure 1: Least Squares Approximation Plot and Dataset Plot

III. ADAM OPTIMIZATION

Adam Optimization (Adam) is a modified gradient descent algorithm. A gradient descent is a process to optimize the cost function. The cost function is the difference between the predicted value and the actual value [3]. For example: Figure 2 is a contour plot of the cost function [3]. The gradient descent follows the downward slope of the system towards the minimum value. The red trace is the path taken by the gradient descent process. Adam Optimization utilizes moments from the first and second gradients to have an adaptive learning rate [5]. The learning rate is simply the step size the system contour. A small learning rate has the tradeoff of requiring more iterations before achieving the minimum value. On the other hand, a large learning rate comes at a risk of missing or jumping over the minimal values [3]. The gradient descent is applied in deep learning and computer learning applications were the dataset is larger enough that Least Squares is not possible.

A. Adam Optimization

Adam Optimization was designed to combine advantages from Adaptive Gradient (AdaGrad) and Root Mean Square Propagation (RMSProp) [6]. Since Adam is an iterative process, the critical equations will be discussed in this section and the code is listed in the Appendix section.

For each iteration, the learning rate is updated through W_t by taking the past iteration into account with a set learning rate, α, and current moment estimations (Equation 12). This ability to adjust the learning rate with minimal memory by only using the previous values and current moment estimates, is what separates Adam from other gradient descent optimization. Equation 8 is a proportional example of the updated learning rate. Equation 9 is the first biased moment estimate. Equation 10 is the second biased moment estimate. To remove those biases, Equation 11 will need to be executed.

$$\frac{average\ gradient}{\sqrt{average\ square\ gradient}} \tag{8}$$

$$M_t = \beta_1 M_{t-1} + (1-\beta_1)\nabla L_t(W_{t-1}) \tag{9}$$

$$R_t = \beta_2 R_{t-1} + (1-\beta_2)\nabla L_t(W_{t-1})^2 \tag{10}$$

$$\widehat{M_t} = \frac{M_t}{1-\beta_1^t}, \qquad \widehat{R_t} = \frac{R_t}{1-\beta_2^t} \tag{11}$$

$$W_t = W_{t-1} - \frac{\alpha \widehat{M_t}}{\sqrt{\widehat{R_t} + \epsilon}} \tag{12}$$

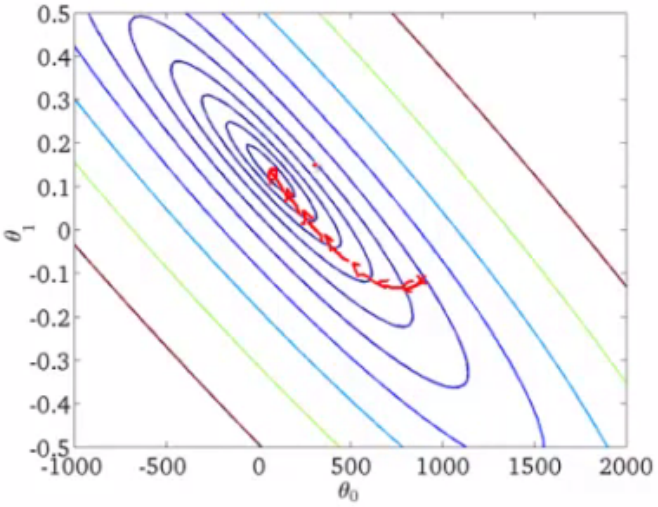

Figure 2: Cost Function Contour Gradient Descent Trace []

B. Setup/Results

Similar to the execution of the Least Squares Approximation methods mentioned before, a single vector was constructed as an input feature for approximation (13)

$$\Theta = \begin{bmatrix} 1 \\ 2 \\ . \\ 275 \end{bmatrix} \qquad \hat{y} = \begin{bmatrix} 5 \\ 11 \\ . \\ 34.5 \end{bmatrix} \tag{13}$$

Constructing a Linear Regression model using Adam Optimizer provided an additional method of testing the input labels to explore other options and algorithms in addition to Least Squares Approximation. Using open sourced python code [4] and libraries such as TensorFlow and NumPy, creating a model to evaluate the label data vectors provided additional methods of evaluating the time series data. The time series vector $\hat{y}$ was split into training and test data for evaluation purposes. From there the setup of the Linear Regression model using Adam was fairly similar to the Least Squares Approximation in defining the model, epochs, and learning rates. Figure 3 shows the output of the Adam model, because some data is used in testing and training the model, the graph is less dense than that of the Least Squares Data Plot in Figure 1.

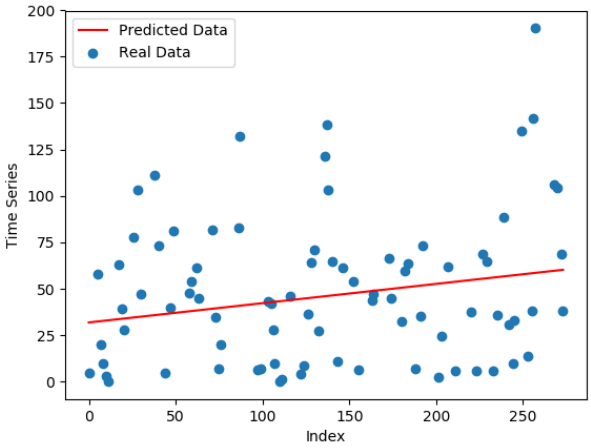

Figure 3: Adam Optimization Plot and Dataset Plot

IV. CONCLUSION

These two methods, Least Squares and Adam, were selected to compare an analytical process to an iterative process to fit a model to a given dataset. The winner will be used to predict the next 30 values, per project instructions. Both methods were relatively simple to execute and generated results in a short amount of time. That was expected since the dataset only consisted of 275 values and only one input feature. A correlation coefficient was calculated for each result and used as a comparison. It is important to note that each method has strengths and weakness. Table 1 lists the strengths and weakness along with the correlation coefficient of each method. According to Table 1, the Least Square Approximation had the highest correlation coefficient.

Table 1: Strengths, Weaknesses, and Results for each Method

	Strength	Weakness	Corr .Coeff
Least	No learning rate No iterations	Large dataset Matrix inverse Redundant features	0.0327
Adam	Large dataset Adaptable learning rate	Many iterations Time/iteration	0.0321

V. SUBMISSION

The following section is the official submission for Project 2 with code and instructions. Figure 4 is the plot of the modified Matlab code. Equation 14 is the hypothesis equation that lists the values of Θ. Table 2 lists the prediction values for the next 30 points.

The load instructions are simple. Open Matlab and place the excel file for the dataset into the file path. Then copy and paste the modified Matlab script and save. Run the Matlab script. The script will output a plot (Figure 4) and save all the predicted values in variable Reg. Once Reg is saved in the workspace, it can be applied to the new dataset to determine error.

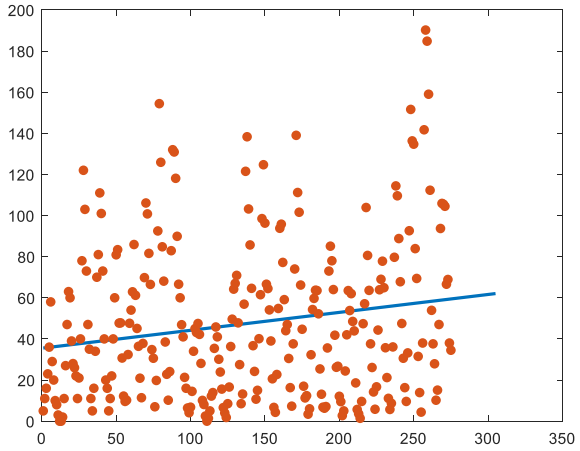

Figure 4: Submission Plot

$$h_\theta(x) = 35.4720 + 0.0871x \qquad (14)$$

Table 2: Submission Prediction Data Points

x1	y1	x2	y2	x3	y3
276	59.52	286	60.3913	296	61.2627
277	59.6072	287	60.4785	297	61.3498
278	59.6943	288	60.5656	298	61.4369
279	59.7814	289	60.6527	299	61.524
280	59.8686	290	60.7399	300	61.6112
281	59.9557	291	60.827	301	61.6983
282	60.0428	292	60.9141	302	61.7854
283	60.13	293	61.0013	303	61.8726
284	60.2171	294	61.0884	304	61.9597
285	60.3042	295	61.1755	305	62.0468

A. Matlab Script for Submission

```
filename = "project2_time series
data_students.xlsx";
data = xlsread(filename);
temp = ones(275,1);
x = [temp data(:,1)];
y = data(:,2);
theta = (x'*x)^(-1)*x'*y
prediction = (1:1:275+30); %Dataset plus
30 prediction points
Reg = theta(1)+theta(2)*prediction;
plot(prediction,Reg,'LineWidth',2);
hold on
scatter(data(:,1),data(:,2),'filled');
```

VI. References

[1] A. A. Rodriguez, "Least Square, Minimum Norm, and Projection Problems," in *Linear Systems - Analysis and Design*, Tempe, Control3D, L.L.C, 2004, pp. 370-372.

[2] G. Strang, "Least Square Approximation," in *Introduction to Linear Algabra*, Wellesley, Wellesley-Cambridge Press, 2009, pp. 218-223.

[3] A. Ng, "Lecture 4.6 - Linear Regression with Multiple Variables | Normal Equation," YouTube, 2016.

[4] R. Norouzy, "Linear regression in tensorflow," Kaggle Inc, Feb 2018. [Online]. Available: https://www.kaggle.com/rohumca/linear-regression-in-tensorflow. [Accessed Sept. 2018].

[5] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference of Learning Representations (ICLR)*, 2015.

[6] N. Cohen, "The Hebrew University of Jerusalem," 18 Oct. 2015. [Online]. Available: https://moodle2.cs.huji.ac.il/nu15/pluginfile.php/316969/mod_resource/content/1/adam_pres.pdf. [Accessed 18 Sept. 2018].

VII. Appendix

A. Least Squares Code

```
1
2     filename = "project2_time series data_students.xlsx";
3     data = xlsread(filename);
4
5     temp = ones(275,1);
6     x = [temp data(:,1)];
7     y = data(:,2);
8     theta = (x'*x)^(-1)*x'*y;
9     Reg = theta(1)+theta(2)*data(:,1);
10    plot(data(:,1),Reg,'LineWidth',2);
11    hold on
12    scatter(data(:,1),data(:,2),'filled');
```

B. Adam Optimization Code

```
1  import matplotlib as mpl
2  mpl.use('TkAgg')
3  import matplotlib.pyplot as plt
4  import tensorflow as tf
5  import numpy as np
6  import pandas as pd
7  from openpyxl import load_workbook
8  from numpy import genfromtxt
9
10 def read_data():
11     training_set =
pd.read_excel('/Users/krystofe/Desktop/timeseries.xlsx',
header=None, names=['features','labels'])
12     features_df =  pd.DataFrame( {'features' :
training_set['features'].values})
13     labels_df = pd.DataFrame( {'labels' :
training_set['labels'].values})
14     features = np.array(features_df.values)
15     labels = np.array(labels_df.values)
16     return features, labels
17
18 def feature_normalize(dataset):
19     mu = np.mean(dataset,axis=0)
20     sigma = np.std(dataset,axis=0)
21     return (dataset - mu)/sigma
22
23 def append_bias_reshape(features,labels):
24     n_training_samples = features.shape[0]
25     n_dim = features.shape[1]
26     f =
np.reshape(np.c_[np.ones(n_training_samples),features],[n_training_
samples,n_dim + 1])
27     l = np.reshape(labels,[n_training_samples,1])
28     return f, l
29
30 features,labels = read_data()
31 normalized_features = feature_normalize(features)
32 f, l = append_bias_reshape(normalized_features,labels)
33 n_dim = f.shape[1]
34
35 #Uses the same data set for testing
36 rnd_indices = np.random.rand(len(f)) < 0.80
37
38 train_x = f[rnd_indices]
39 train_y = l[rnd_indices]
40 test_x = f[~rnd_indices]
41 test_y = l[~rnd_indices]
42
43 learning_rate = 0.01
44 training_epochs = 1000
45 cost_history = np.empty(shape=[1],dtype=float)
46
47 X = tf.placeholder(tf.float32,[None,n_dim])
48 Y = tf.placeholder(tf.float32,[None,1])
49 W = tf.Variable(tf.ones([n_dim,1]))
50
51 init = tf.global_variables_initializer()
52
53 y_ = tf.matmul(X, W)
54 cost = tf.reduce_mean(tf.square(y_ - Y))
55 training_step =
tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
56
57 sess = tf.Session()
58 sess.run(init)
59
60 for epoch in range(training_epochs):
61     sess.run(training_step,feed_dict={X:train_x,Y:train_y})
62     cost_history =
np.append(cost_history,sess.run(cost,feed_dict={X: train_x,Y:
train_y}))
63
64 plt.plot(range(len(cost_history)),cost_history)
65 plt.axis([0,training_epochs,0,np.max(cost_history)])
66 plt.show()
67
68 pred_y = sess.run(y_, feed_dict={X: test_x})
69 mse = tf.reduce_mean(tf.square(pred_y - test_y))
70 print("MSE: %.4f" % sess.run(mse))
71
72 fig, ax = plt.subplots()
73 ax.scatter(test_y, pred_y)
74 ax.plot([test_y.min(), test_y.max()], [test_y.min(), test_y.max()],
'k--', lw=3)
75
76 ax.set_xlabel('Measured')
77 ax.set_ylabel('Predicted')
78 plt.show()
```