**Linear Prediction with Real Data**

Owen Sowatzke

Georgia Institute of Technology

School of Electrical and Computer Engineering

ECE 4271

Spring 2021

**Introduction**

The goal of this project was to design and evaluate linear predictors using MATLAB. Specifically, this project examined linear predictors in applications involving the stock market and Ethereum block difficulty. This document details the results of each of these analyses and is divided into two sections that investigate prediction of the stock market and Ethereum block difficulty respectively.

**Part I**

The linear prediction of the stock market that follows parallels exercises in the Buck et al. textbook. 91 years of weekly stock market data from Oct. 1, 1928 to Feb. 11, 2019 is used to design linear predictors and then analyze the resulting predictions.

**(a)**

The Dow Jones Industrial Average Data is plotted on a linear and semi-logarithmic scale in Figures 1 and 2 respectively.

Chart

Description automatically generated

Figure 1 – Plot of the Dow Jones Industrial Average on a Linear Scale.

Chart, line chart

Description automatically generated

Figure 2 - Plot of the Dow Jones Industrial Average on a Semi-Logarithmic Scale.

Assuming that we start with $1000 and invest all our money in the DJIA, we would have $104,196.93 at the end of investment interval (4714 weeks). If we decided to instead put all our money in the bank at %3 APR, it would take 8056 weeks to make the same amount of money. If we wanted to make the same amount of money over the same time interval, we would need a 5.13% APR.

**(b)**

Using and, we can solve for the vector that minimizes the inner product:

. Using the MATLAB’s operator, we get the following results:

**(c)**

Using and the command, we can predict values for the first decade of decade. Figure 3 displays the predictions on the same set of axes as the actual weekly average.

Chart, line chart, histogram

Description automatically generated

Figure 3 – Plot of the Predicted and Actual Weekly DJIA Data over the First Decade.

Using , we can calculate the total squared error according to the following formula: . This gives us the following result:

To check the results, we can compute according to the following formula: . Then, we can calculate using the formula above. Doing so, gives us the following result:

**(d)**

In Figure 4, we plot the total squared prediction error as a function of for .

Chart, line chart

Description automatically generated

Figure 4 – Plot of the Total Squared Error vs for .

There appears to be a “knee” at . However, the total squared prediction error drops steeply starting at . Therefore, there is not a value of after which the decrease in prediction error is negligible. As such, we choose because it minimizes the prediction error.

**(e)**

If we have $1000 at the start of the p-th week and make 520 trading decisions, the upper bound to how much we could make is $4,700,565.98 - $1000 = $4,699,565.98. One lower bound to how much we can make is the amount we can make if all the money is invested in the bank. If this is the case, we can make $1349.74 - $1000 = $349.74. Another lower bound to how much we can make is the amount we can make if all the money is invested in the DJIA. If this is the case, we can make $544.44 - $1000 = - $455.56. Using the linear predictor, we can make $1422.65 - $1000 = $422.65. This is equivalent to an APR of 3.53%.

**(f)**

Using the linear predictor coefficients derived in (e), we can use the same prediction strategy on the most recent decade of data. The upper bound to how much we can make is $104,798.92 - $1000 = $103,798.92. By investing the $1000 in the bank, we can determine one of the lower bounds, which is $1349.74 - $1000 = $349.74. If we instead invest the $1000 in the stock market, we can determine the other lower bound, which is $3521.28 - $1000 = $2521.28. Using our linear predictor, we can make $2265.53 - $1000 = $1265.53. This is equivalent to an APR of 8.18%.

**Part II**

In this part, we use the autocorrelation method to predict Ethereum block difficulty.