Supervised Learning for Linear Prediction (Auto-regression on stock market data using LMS & RLS)

Yashad Samant
M.S. (Electrical & Computer Engineering)
Colorado State University

***Abstract*— The objective of this document is to discuss the supervised learning method (auto-regression) used for Linear Prediction of stock market. We select data which is time-variant, observe the features of the data using time series analysis and apply regression techniques on the data to predict the future outcomes of the data. Our second objective is to minimize the error using two methods – Least Mean Squares and Recursive Least Squares.**

***Keywords* – LMS, RLS, regression, supervised, time-series analysis.**

I. Introduction

In this paper, our main objective is to predict the next sample of a time variant data using supervised learning method. To achieve this, we have used auto-regression method which incorporates the time variant samples very well. We have designed a system for N=3 which means we are training the data on the basis of three previous samples.

Thus, to start with we obtained the data from [1] which is Apple's logged historical data for the past three years. Data consisted of approximately 1300 samples in seven different samples, where one was the logged date.

Our second objective was to generate a auto-regression model with N=3 where the error is minimized using Least-Mean-square algorithm. We generated a generalized model instead with N=3 so that we can use it with any data.

Our third objective was to change the step size of the LMS algorithm to 10^-2, 10^-3, 10^-4 and check if there is the results.

Our fourth objective was to validate the data and calculate the Mean-Squared-Error to check the performance of the predictor and see if the predictor is reliable enough.

Last objective was to test the data on the test data set and check if the output is correct making it a good predictive model.

Block diagram of our model is as follows:

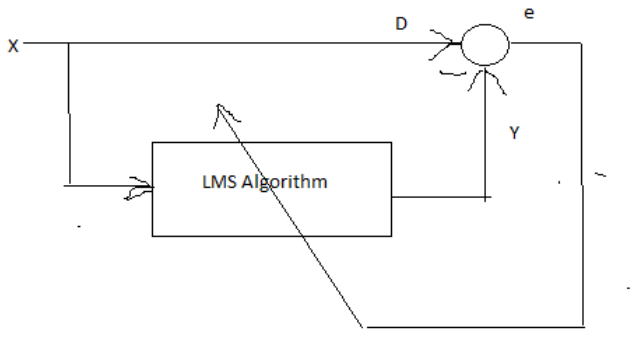


Fig 1: Block Diagram

II. Theory

In this section, we are going to get insight on the data, LMS & RLS algorithm

A. Data

Data was extracted from [1] which consists Apple's Stock data for the past three years. Data consisted of seven columns with 1259 samples of each to be precise. The column names were: Date, Open, High, Low, Close, Adj Close and Volume. Volume and date were irrelevant to my prediction and hence we didn't use it. All the samples were in the range of 40 to 200 and probably for this data as the range was so close standardization wouldn't have been necessary but doing it a is a good practice as relative values are very easy to calculate and normalized data has zero mean thus plotting the distribution of data becomes easier. Relevant data is as follows:

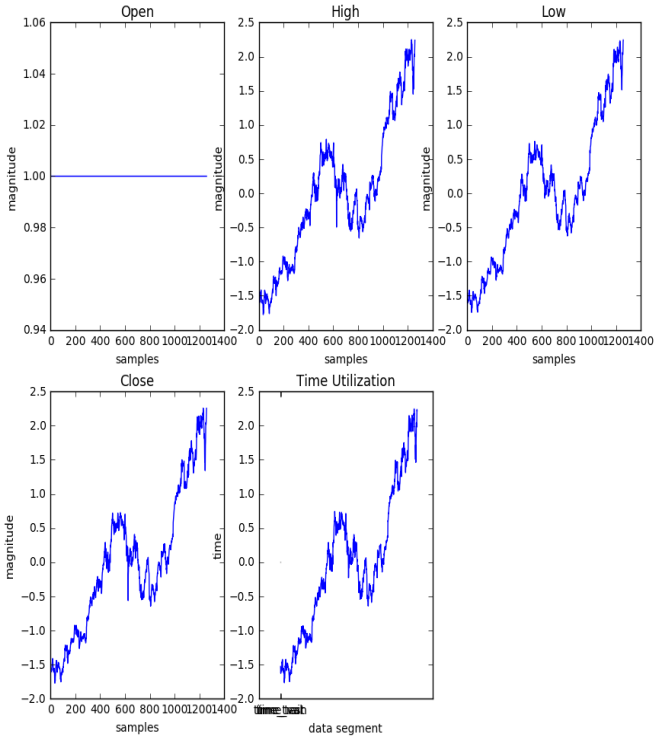


Fig 2: Data

Also, as we can see in the plot above that OPEN has more or less constant values and hence removing OPEN is a good idea. It won't affect the output much but it doesn't contribute in the prediction, hence the data is redundant. Also, we have simulated the model on an easier example but in real world we have n-dimensional data with millions of samples. Thus, removing obsolete data is a good practice as it fulfils couple of motives: 1) Reducing the dimensionality of the data thus lowering down the process time. 2) Lowering any chances of over-fitting the model.

Thus, we achieved various objectives in this case.

- Extracted the data in a data frame format using pandas.
- Normalized the data to avoid any error due to high weights to high valued data.
- Got rid of any obsolete or redundant data.

B. *Least-Mean-Square Algorithm*

LMS is method where we update the weights to achieve the least mean square between the input and the output. We can also see that in Fig 1.

The basic idea behind LMS filter is to approach the optimum filter weights (R-1 P), by updating the filter weights in a manner to converge to the optimum filter weight. The algorithm starts by assuming small weights (zero in most cases), and at each step, by finding the gradient of the mean square error, the weights are updated. That is, if the MSE-gradient is positive, it implies, the error would keep increasing positively, if the same weight is used for further iterations, which means we need to reduce the weights. In the same way, if the gradient is negative, we need to increase the weights. So, the basic weight update equation is:

$$\mathbf{wn+1 = wn - \mu\Delta\varepsilon[n]}$$

Where, $\boldsymbol{\varepsilon}$ represents the mean-square error. The negative sign indicates that, we need to change the weights in a direction opposite to that of the gradient slope.

For this example, we started by initializing the filter length, size of desired output and size of the error matrix. Iterated it over the number of training samples to achieve the required result. We took the filter length to be 3 as the output was based on three samples in time, where values of w were multiplied with their respective xs. Thus, each time sample had just one weight to change.

We also changed the size of w and saw that larger the filter length better was the output.

C. *Recursive Least Squares*

The Recursive least squares (RLS) adaptive filter is an algorithm which recursively finds the filter coefficients that minimize a weighted linear least squares cost function relating to the input signals. The RLS algorithms are known for their excellent performance when working in time varying environments but at the cost of an increased computational complexity and some stability problems.

We implemented RLS using pdasift framework in python. It was a direct implementation and we just had to insert the values of x and y to achieve the output.

D. *Comparison: LMS V/S RLS*

TABLE I

Sr.No	Algorithms	MSE	Complexity	Stability
1	LMS	$1.5*10^{-2}$	2N+1	Less
2	RLS	$6.2*10^{-3}$	$4N^2$	High

Thus, we can see that RLS is much better than LMS. We will discuss the comparisons in detail in the Result section.

III. RESULTS

In this section, we discuss the results of the LMS code.

A. *Data*

We have discussed the data section in detail where we achieved the following objectives.

- Extracted the data in a data frame format using pandas.
- Normalized the data to avoid any error due to high weights to high valued data.
- Got rid of any obsolete or redundant data.
- Dividing the data into three sections train (0.5), validate (0.25) and test (0.25).

B. *LMS algorithm*

Train data had half the data which comes out to be about 630 samples. This data was given to the LMS algorithm to train and update the weights of the autoregression model. Results obtained from train data are as follows:

a. Trained Output

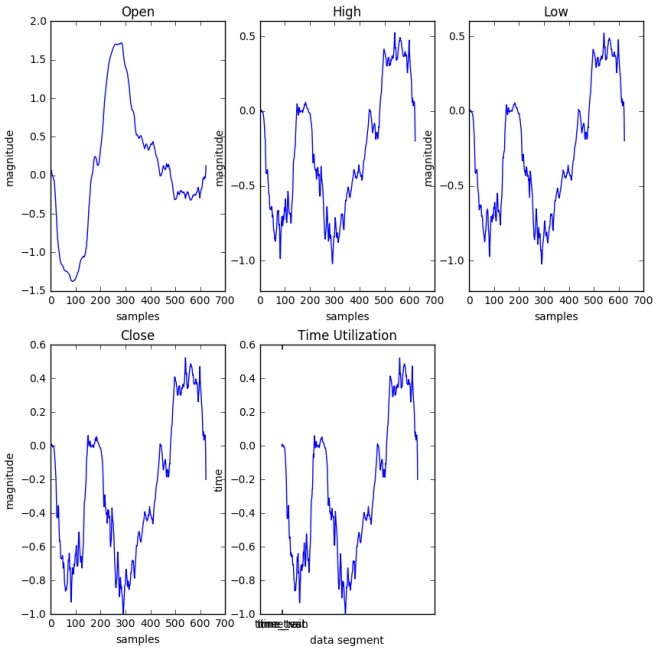


Fig 3: Trained output

Trained data is plotted and we can see the results matching the resultant data. We can also see that we got oscillations in the OPEN section where we should have obtained a straight line. It's because of the over-fitting problem we discussed earlier. If we don't include OPEN in the data, result is much more crisp and accurate.

Test and validation data were given quarter data each making it 310 samples for each set. The results for both the data sets can be seen below:

b. Validate Output

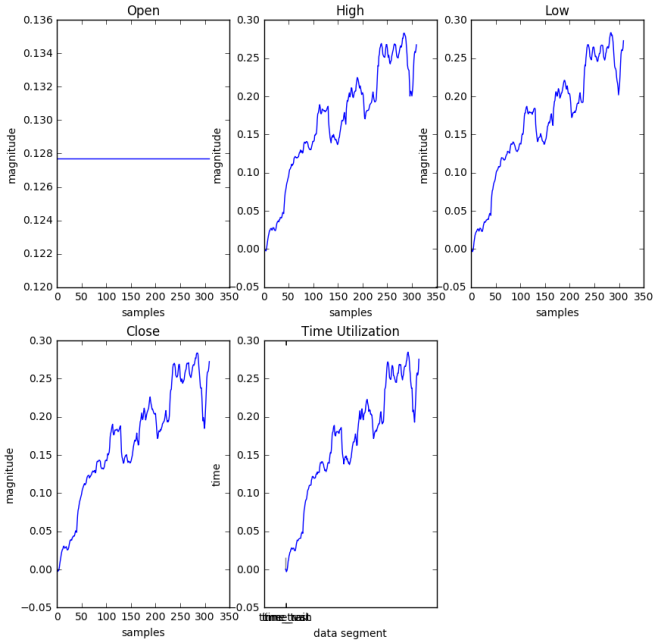


Fig 4: Validation output

We received the highest error for validation data set. It might be because of getting stuck in local minima. We can counter this problem by using k-fold method where we can start the problem with multiple starting points to find the optimal result.

c. Test Output

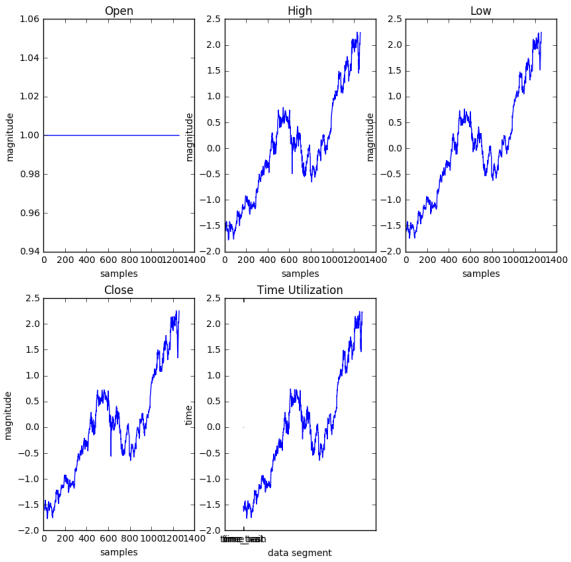


Fig 5: Test output

We can see our model performed very well on the test data-set giving very low error.

d. *Mean-Squared-Error*

We calculated the Mean-Squared Error between actual and desired output for train, test and validate data samples. This is how they performed. We can see that the model was well trained giving just error for test data-set where it performed better than train data-set. It didn't perform well for validation data-set though. Probably, as it started from different starting point, it got stuck in the local minima. We can avoid this problem by using k-folds method where we fine tune the parameters using different sets of validation data sets and obtaining the optimal result.

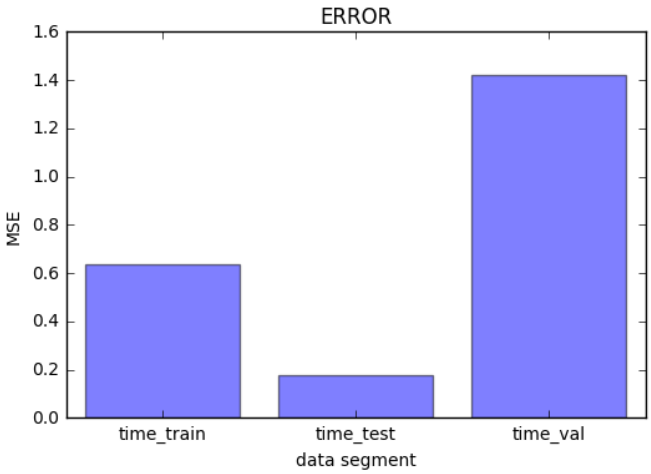


Fig 6: MSE Error

e. *Time Utilization*

Time utilization for each data-set was also calculated where complementing the Table 1 LMS took 0.015 seconds to converge for train data-set while validate and test data-set took negligible time.

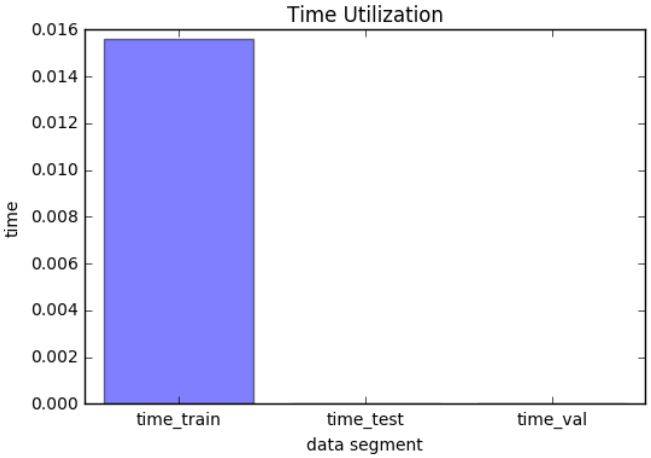
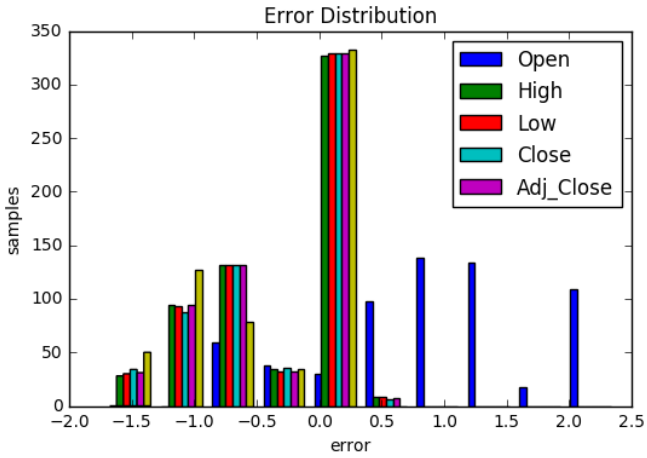


Fig 7: Time Utilization

f. Error Distribution

We have plotted the error distribution over various samples in the test output.



C. Comparison RLS V/S LMS

RLS algorithm was also implemented to develop comparisons between LMS and RLS and also to check the relative performance of LMS with respect to RLS.

RLS was implemented using padasip module in python. We obtained the following results:

a. Error

The error obtained by RLS method was 0.06 on the training output which is significant improvement on LMS

IV. Conclusions

Thus, we were able to implement supervised learning model from scratch where we minimized the error using both LMS and RLS algorithm without any modules and framework. We obtained the following objectives:

- Extracted the data in a data frame format using pandas.
- Normalized the data to avoid any error due to high weights to high valued data.
- Got rid of any obsolete or redundant data.
- Dividing the data into three sections train (0.5), validate (0.25) and test (0.25).
- Obtained a train, test and validation error of 0.6, 0.015 and 1.4 respectively. We can increase the accuracy by increasing the N of the model or taking lot of past samples.
- Achieved 80% accuracy on the regression model.

Acknowledgment

I wish to acknowledge Dr. Azimi for showing us the statistical picture of Machine Learning and also for enormous help provided with the computer assignment.

References

[1] https://finance.yahoo.com/quote/AAPL?p=AAPL

[2] ISSN: 2278 – 7798 International Journal of Science, Engineering and Technology Research (IJSETR) Volume 2, Issue 5, May 2013J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[3] https://wiki.python.org/moin

[4] https://pypi.python.org/pypi/padasip

[5] https://dsp.stackexchange.com/questions/8184/relative-performance-of-rls-and-lms-filters

[6] *http://www.cs.tut.fi/~tabus/course/ASP/LectureNew10.pdf*.