Machine Learning Project Report - Neural Granger Causality for Nonlinear Time Series

Group: Neural Fatality
Members: Vaibha Sharma (vs3br), Murugesan Ramakrishnan(mr6rx)

April 30, 2019

1 Abstract

Granger causality is an issue where there exists a causal relationship between two or more different time series data. Complex data sets involving stock market has higher probability of such problems since the stock price of one company can be highly dependent on the rise and fall of its competitor. In this paper we have delved deep to identity non-linear causality between such companies using deep neural network. There are two main implementations done *i) Feed Forward network with Group Lasso and ii) LSTM network with Group Lasso*. Each of the above architecture is designed in such a way that the weights of competitor stocked can be interpreted and checked if they are significant in the final predicted value. Analyzing relations between EBay, Amazon, Apple for predicting the stocks of Facebook we able to detect non-zero stock values of the competitors.

Final architecture for Feed forward network contained one hidden layer with 50 units and separate weights (for Group Lasso) for different input stocks. The final LSTM model had one hidden layer with 10 units with would taken in up to 10 lag values to determine the final prediction.

2 Introduction

The overall objective of the project is to identify Granger Causal relationship between different time series data X and Y. Granger causality is when a variable X that evolves over time Granger-causes another evolving variable Y. More precisely, it is when the prediction of Y is based on not just its own past values, but also on the past values of X.

The current use case is to predict the stock prices for a company based on the past values of the same and different company's stock using non-linear methods like Neural Network. Facebook, Amazon and E-Bay's stock are analyzed to identify if there is any non-linear granger causuality using both Feedforward and LSTM methods.

3 Literature Review - Prior Works

Alex Tank et al's work using Multi Layered Perceptron (MLP) and LSTMs have analyzed how deep learning could be leveraged for Granger causality. Our current work aims to introduce similar architecture using MLP and recurrent neural for the use case of stock prices. Here is a brief review of their MLP approach,

A simple two layer MLP is built with the first layer being our time series of interest i, and another time series data j which might potentially cause causality. With a single hidden layer the next value in our time series is predicted. Once the model is built with Residual sum of squares as the loss function, the weight matrix for the input layer is checked. If there are non-zero weights for the time series y, then there is a possibility of Granger causality. This method using MLP is tried for all other possible time series using a group lasso loss to detect the presence of Granger causality. The disadvantage with this approach is that lag value K has to be selected manually. The following equation is used as the MLP optimization problem with a hierarchical group lasso penalty

$$\min_{\mathbf{W}} \sum_{i=1}^{T} \left(x_{it} - g_i(x_{(t-1):(t-K)}\right)^2 + \lambda \sum_{j=1}^{p} \sum_{k=1}^{K} \|(W_{:j}^{1k}, \ldots, W_{:j}^{1K})\|_F$$

4 Data

The data consists of stock prices (First/Last/High/Low Trade Prices) for different companies like Amazon, Facebook, E-Bay, Costco, Twitter, LUV among others. The prices are recorded at a minute level every day from 2013-2018. 2018 data been used for the current Phase 2 analysis which will further be scaled to have all the years for the final phase. One of the problems with the dataset is that many of the company's do not have consistent recorded price, but have a lot of *NA* values in them. This has been handled in the Pre-processing step. Here is the split of train/test for the current analysis.

Data	Time Period	Number of Records
Train	2018/01/02 - 2018/03/09	3000
Test	2018/03/10 - 2018/03/29	904

There is also another data source with the tweets about companies data at a minute level. Each tweet has been associted with a sentiment which might help in predicting the change in stock prices. Although, the tweets were not used as a part of the input for the Phase 2, it would be implemented for the final phase.

5 Pre-processing step

5.1 NA Imputation

Each of the NA values were imputed with the average of the previous and the next value

5.2 Minute Base - Window

To create a consistent time window for the stock prices, the minute data was converted into a widow of 15 minute each where the data was aggregated to have the maximum value found in each window. Last Trade Price was taken as the main KPI (Key Performance Indicator) for further analysis.

5.3 Lag value calculation

Different lag value K is calculated for each stock. K=5 (lag 5) is determined to be optimal based on the results of preliminary Machine Learning algorithms. This is used for detecting Granger causality using Machine Learning and MLP methods

5.4 Data Transformation

Having selected Last Trade Price as the KPI, this information is aggregated across all the day's data for entire 2018 for all the companies. Finally, a clean data with all the company's prices is generated which looks as follows,

Date	Hour	Minute_15	FB_LastTradePrice_lag0	EBAY_LastTradePrice_lag1
20180102	5	0	176.4	1167.92
20180102	5	15	176.4	1170
20180102	5	30	176.45	1170
20180102	5	45	176.48	1170
20180102	6	0	176.65	1170

This final prepared data is used for Machine Learning and Deep Learning models.

6 Baseline model description

6.1 Two Stocks

Different pairs of stock prices Facebook vs. Amazon, Amazon vs. EBay, Ebay vs. Apple are tried to see if any granger causality exists between the stocks. A simple linear regression Model is tried to identity two factors that affect the final prediction,
1) Number of lag values
2) See the significant coefficients (Check if the other company's stock play a role)

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     FB_LastTradePrice_lag0   R-squared:                       1.000
Model:                                OLS   Adj. R-squared:                  1.000
Method:                     Least Squares   F-statistic:                 7.848e+07
Date:                    Wed, 27 Mar 2019   Prob (F-statistic):               0.00
Time:                            00:53:30   Log-Likelihood:                -2409.0
No. Observations:                    3904   AIC:                             4834.
Df Residuals:                        3896   BIC:                             4884.
Df Model:                               8
Covariance Type:                nonrobust
================================================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
FB_LastTradePrice_lag1         1.1682      0.017     70.339      0.000       1.136       1.201
FB_LastTradePrice_lag2        -0.2323      0.025     -9.124      0.000      -0.282      -0.182
FB_LastTradePrice_lag3         0.0616      0.025      2.419      0.016       0.012       0.111
FB_LastTradePrice_lag4         0.0024      0.017      0.145      0.884      -0.030       0.035
EBAY_LastTradePrice_lag1       0.2904      0.062      4.668      0.000       0.168       0.412
EBAY_LastTradePrice_lag2      -0.1735      0.086     -2.011      0.044      -0.343      -0.004
EBAY_LastTradePrice_lag3      -0.2209      0.086     -2.559      0.011      -0.390      -0.052
EBAY_LastTradePrice_lag4       0.1047      0.062      1.684      0.092      -0.017       0.227
==============================================================================
Omnibus:                     1952.715   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           335417.307
Skew:                           1.296   Prob(JB):                         0.00
Kurtosis:                      48.335   Cond. No.                     5.78e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.78e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

From the above results for Facebook vs. Ebay, 5 lag variables was found to be effective and it can be clearly seen that lag values of Ebay affects Facebook's prediction - hence proving a granger causual relationship.

This is further tried with Neural Network models.

6.2 Multiple Stocks

Another baseline with more than two stocks was created so that it would be easier to compare with the LSTM model results. The results are,

OLS Regression Results

Dep. Variable:	FB_LastTradePrice_lag0	R-squared:	1.000
Model:	OLS	Adj. R-squared:	1.000
Method:	Least Squares	F-statistic:	3.149e+07
Date:	Mon, 22 Apr 2019	Prob (F-statistic):	0.00
Time:	19:05:07	Log-Likelihood:	-2397.1
No. Observations:	3904	AIC:	4834.
Df Residuals:	3884	BIC:	4960.
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
AMZN_LastTradePrice_lag1	-0.0033	0.002	-1.348	0.178	-0.008	0.002
AMZN_LastTradePrice_lag2	0.0034	0.003	0.971	0.332	-0.003	0.010
AMZN_LastTradePrice_lag3	-0.0025	0.003	-0.708	0.479	-0.009	0.004
AMZN_LastTradePrice_lag4	0.0019	0.002	0.782	0.434	-0.003	0.007
COST_LastTradePrice_lag1	-0.0030	0.019	-0.160	0.873	-0.040	0.034
COST_LastTradePrice_lag2	0.0120	0.025	0.478	0.633	-0.037	0.061
COST_LastTradePrice_lag3	0.0101	0.025	0.401	0.688	-0.039	0.059
COST_LastTradePrice_lag4	-0.0171	0.019	-0.905	0.366	-0.054	0.020
CSCO_LastTradePrice_lag1	-0.1298	0.071	-1.824	0.068	-0.269	0.010
CSCO_LastTradePrice_lag2	0.2199	0.093	2.353	0.019	0.037	0.403
CSCO_LastTradePrice_lag3	-0.1117	0.094	-1.193	0.233	-0.295	0.072
CSCO_LastTradePrice_lag4	0.0289	0.071	0.406	0.685	-0.111	0.169
EBAY_LastTradePrice_lag1	0.3388	0.065	5.207	0.000	0.211	0.466
EBAY_LastTradePrice_lag2	-0.2406	0.090	-2.680	0.007	-0.417	-0.065
EBAY_LastTradePrice_lag3	-0.1798	0.090	-2.000	0.046	-0.356	-0.004
EBAY_LastTradePrice_lag4	0.0964	0.065	1.483	0.138	-0.031	0.224
FB_LastTradePrice_lag1	1.1819	0.018	65.188	0.000	1.146	1.217
FB_LastTradePrice_lag2	-0.2537	0.028	-9.219	0.000	-0.308	-0.200
FB_LastTradePrice_lag3	0.0711	0.028	2.584	0.010	0.017	0.125
FB_LastTradePrice_lag4	-0.0027	0.018	-0.147	0.883	-0.038	0.033

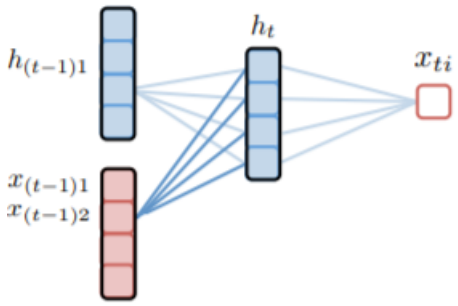
Omnibus:	1900.528	Durbin-Watson:	1.996
Prob(Omnibus):	0.000	Jarque-Bera (JB):	312311.712
Skew:	1.241	Prob(JB):	0.00
Kurtosis:	46.747	Cond. No.	5.20e+04

From this linear regression results it can be seen the Facebook's stock is mainly dependent on E-Bay's past values. Amazon, Costco or Cisco does not have any linear relationship with Facebook's future stock.

7 Method/analysis

7.1 MLP with Group Lasso

The input data is split into two - Current Company's Stock and Competitor Company's stock and each of them are fed into a hidden layer separately (H11 and H12). This is concatenated to form a single hidden layer H1 (Ht in the image) which is then passed again another optional layer H2 followed by the output Y (xti in the below image). This can be seen from the below architecture,



The loss function would be based on group LASSO method and can be

represented as,

$$\min_{\mathbf{W}} \sum_{t=K}^{T} \left(x_{it} - g_i(x_{(t-1):(t-K)}\right)^2 + \lambda \sum_{j=1}^{p} \|(W_{:j}^{11}, \ldots, W_{:j}^{1K})\|_F.$$

The idea is to check the weights W11 and W12 which is a part of the hidden layer H11 and H12. If the weights W12 (Weights correspond to the competitor company) are non-zero then there is a granger causality between the two company's stock prices.

7.2 MLP : Final Model Architecture

Input Layer: The input is split into two - current company stocks X1, other stocks X2

Hidden Layer 1:

H11: No of Units 5
H12: No of Units 5
H1: No of Units 10

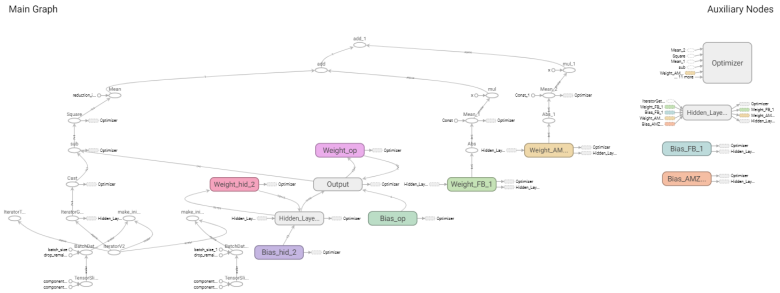
H11 = ReLU(W11 * X1 + B11)
H12 = ReLU(W12 * X2 + B12)

H1 = concatenate(H11, H12)

Hidden Layer 2
No of units: 15
H2 = ReLU(W2 * H1 + B2)

Output Layer
Y = W3 * H2 + B3
This Y is considered as the final output.

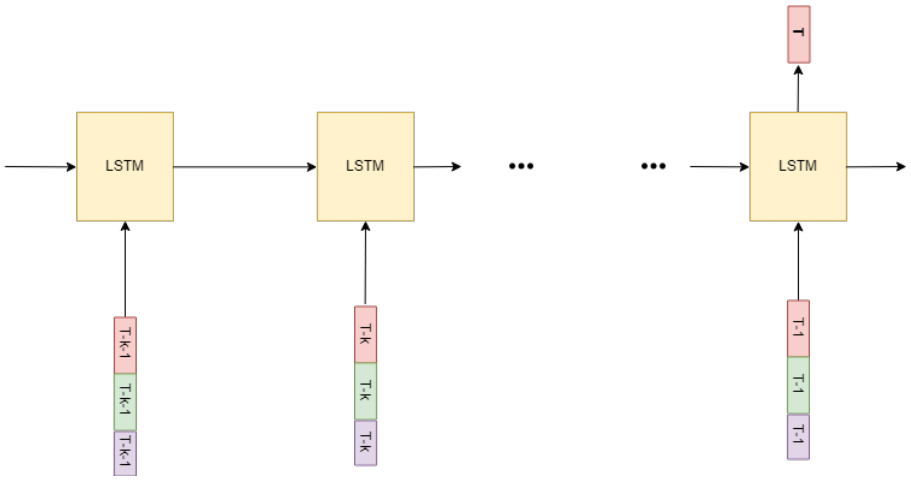
Following has been obtained from Tensor Board,



7.3 LSTM Implementation with Group Lasso

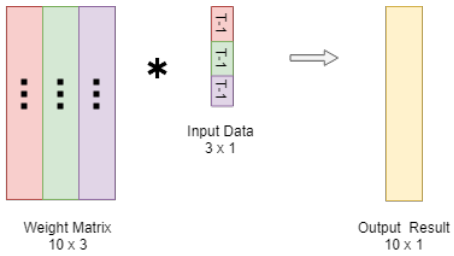
The objective is to predict the Facebook's stock price at time T. The input has different historic time units of competitors with up to a maximum K lags so that the LSTM model could identify in itself how much of the lag values are important for the final prediction.

Each unit is fed with the lag values of different companies's stocks as shown,



For instance Red represents Facebook, Green represents E-Bay and Purple represents Amazon. The input is fed in such a way because this would help in interpreting weights for each of them specifically from the hidden unit weights.

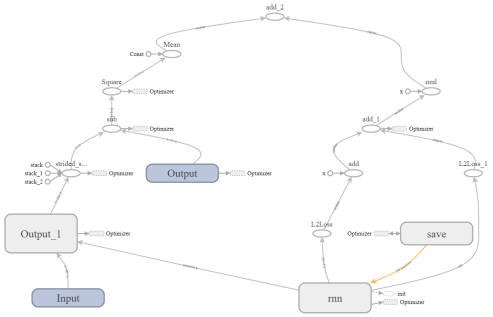
For a hidden unit size 10, the weight matrix would look as

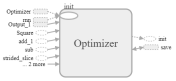


Say analyzing the values of green column would represent the importance for E-Bay stock prices. For proper interpretability, group Lasso loss function was implemented so that insignificant weights would tend to become zero. The loss function had two parts, i) Mean Squared Error between the final prediction and actual value ii) L1 distance of weights in the 10x3 Matrix. The overall loss function is similar to the loss used for MLP above.

7.4 LSTM Implementation - Final Architecture

The final architecture used for prediction as obtained from Tensor board is shown as,





The specification of final architecture is as follows,
Number of total time steps: 10
Number of hidden units H1: 5
Dimensions:
Initial hidden state H0: 10 x 1 (Random Initialization)
Input: 5 x 1
LSTM Unit:
H1 = tanh(W1*X + W2*H0 + bias)
Output = (W3 * H1 + bias)

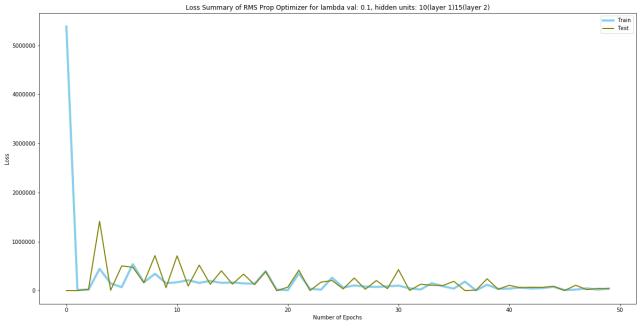
This output at the last LSTM unit is used for loss calculation

8 Result analysis

8.1 Multi-Layered Perceptron

The model was tried with different epoch values from 50 to 1000, and optimal values were seen for epoch 50 beyond which the loss started to increase.Also, learning rate from 0.001,0.01,0.1,0.2 was tried for checking how the values converge

The plot is an example for training and test error. Although there is fluctuation for first few epochs, the error rate is convereged



Here is a small analysis on how the train and test errors were at the end of 50 epochs for different learning rate.

Learning Rate	Train Loss	Test Loss
0.001	3594.8403	18379.28
0.01	180.125	132.4917
0.1	80.74	223.22
0.2	83.26	221.26

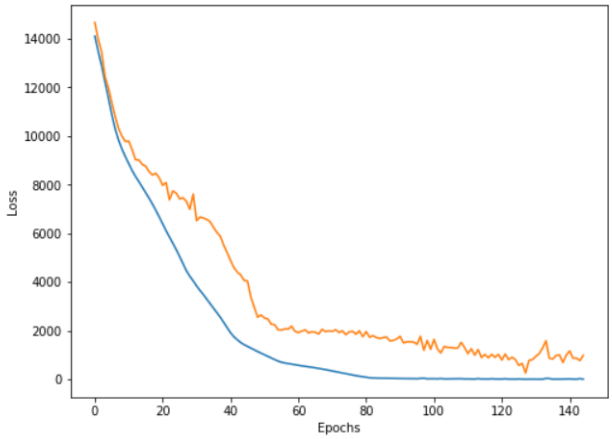
Learning rate of 0.01 was found to be the best. Further analysis on various other architectures are given below.

Most importantly, the W11 and W12 Matrix were analyzed to check if all of the W12 were zero. Looking at the values, the Competitor share's were non-zero confirming there is a**Granger Causal relationship**

```
In [26]: print('FB weights')
         sess.run(W11)

         FB weights

Out[26]: array([[ 1.0435531e+00, -1.7221256e-01, -4.1623056e-01,  1.2192260e-01,
                   1.5734999e+00,  1.9701408e+00,  5.0107843e-01, -3.5803670e-01,
                  -1.7452171e-01, -2.2636386e-02],
                 [-1.0802858e+00, -3.0547023e-01, -2.0818507e-02,  5.6480306e-01,
                   1.7582814e+00, -1.7808981e+00,  9.7767301e-02,  7.7420765e-01,
                  -1.0431775e+00, -7.9371384e-06],
                 [ 1.8524036e+00,  9.1498047e-01,  4.7353986e-01,  9.8456615e-01,
                  -1.1205171e+00, -7.0923561e-01, -3.3924595e-01, -4.0901071e-01,
                  -1.1034880e+00, -2.4920616e-02],
                 [-1.3356547e+00,  1.3956523e+00,  6.1864704e-01,  1.8853475e-01,
                   1.2965270e+00,  4.9572447e-01, -3.6622170e-01,  8.9676279e-01,
                   1.2124494e+00, -2.4844259e-02]], dtype=float32)

In [27]: print('EBAY weights')
         sess.run(W12)

         EBAY weights

Out[27]: array([[-0.02293844,  0.0455966 ,  1.3657751 ,  2.3138704 ,  0.32384297,
                  -0.14572893,  0.05250383,  0.84557986, -0.19552238,  0.40646467],
                 [ 2.4104443 ,  0.21704322, -0.39032754,  0.5500849 , -0.6792396 ,
                  -0.03815933,  0.10660086,  1.0408696 , -1.4174595 ,  0.17883243],
                 [ 0.942521  ,  1.2008336 , -0.12593664,  0.5985514 ,  1.4071366 ,
                  -0.24515964,  0.04504711,  1.2650261 , -0.12647524, -0.40296987],
                 [ 0.19518276,  0.679167  , -0.40602714, -0.41725454,  0.5692248 ,
                   0.420968  , -0.2059299 , -1.6067717 ,  1.8568673 , -0.17924803]],
                dtype=float32)
```

8.2 LSTM Method

The model was trained at various epochs up to 1000 and it was found that the loss value saturates at 100 although the validation loss fluctuates slightly after that.



Key: Orange = 'Validation', Blue = 'Training'

Different learning rates and lambda (weights) values for the loss function were tried to find the optimal value for faster convergence. The lambda value was set to 2000 so that the weights converge faster to zero, and the learning rate of 0.01 helped in better optimization without exploding or vanishing gradients problem.

For different learning rates, it was seen that

After selecting the optimal parameters, the weights were analyzed to check the Granger relationship,

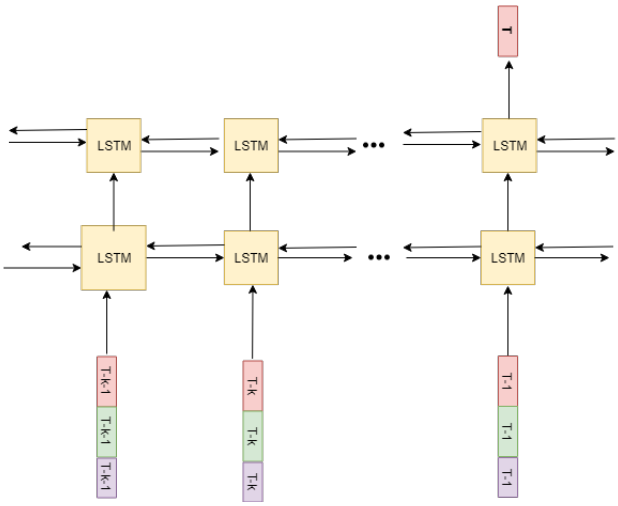
```
Weights Analysis

Facebook:
[[ 9.80658806e-05  1.54984042e-01  1.26751520e-05 -2.38275051e-01
  -7.40835530e-05  2.32832535e-05 -1.78460255e-01 -1.12349997e-04
  -2.01579514e-05 -3.07680457e-05]
Amazon
 [ 2.25591793e-06  7.89419573e-06 -2.24999749e-05  8.12659491e-05
  -1.35819660e-04 -1.56326030e-04  1.37614465e-04 -5.69014956e-05
   1.00067700e-06  1.24344806e-05]

E-Bay
 [ 2.94486791e-01 -6.27119616e-02  2.09584758e-01  6.10653646e-02
  -2.66809511e-05  4.72588226e-06  4.35210884e-01  1.12600064e-05
  -8.56702551e-02 -2.01670355e-06]
Costco
 [-5.82691973e-05 -3.77750993e-02  1.10125951e-02  1.58948887e-06
   3.87359396e-05 -5.96285499e-05 -1.54012829e-01 -1.00555771e-04
  -5.30902253e-05  1.78813425e-05]
Cisco
 [-1.15632618e-04  1.94551121e-05  1.37871539e-04  3.47278874e-05
   8.61279987e-05 -6.24389868e-05 -3.45325680e-05 -6.49481080e-07
  -4.53781977e-05  6.50834118e-05]
```

For predicting the Facebook's stocks, it was compared with the lag prices of itself, E-Bay, Amazon, Costco and Cisco. Weights of greater than 0.01 was set as the threshold for selecting significant stock. As it can be seen Amazon and Cisco do not have any Granger causal relationship with Facebook's stocks. However, E-Bay and Costco has non-linear Granger causal relationship with Facebook.

Although from the baseline Machine Learning only E-bay was found to be a significant stock, LSTM model shows that Costco's prices also help in predicting future price of Facebook. This confirms that there is a non-linear relationship between Costco and Facebook which has been identified by our LSTM model.

9 Additional Implementation - LSTM

1) Apart from required uni-directional LSTM of one hidden layer, we also tried bi-directional LSTMS with multiple layers. Following is the architecture,



Since the model was complicated it was difficult to interpret the weights since now there would be two sets of weights - one for each direction. Further, the additional number of parameters caused more time for the model to converge, and also making the loss function different which made it difficult to compare to basic LSTM model.

We believe these constraints lead the model to not be the best approach for current use case.

10 Improvement - MLP Method

10.1 Regularization

Lasso Regularization was applied to the loss function with different values of lambda 0,0.1,0.5,0.9 to check the overall model performance

Lambda	Train Loss	Test Loss
0	134.56	234.67
0.1	280.125	232.4917
0.5	350.15	325.18
0.9	543.23	567.36

Based on the results, lambda value of 0.1 is selected for further analysis

10.2 Model Size and Layers

Different Hidden Unit sizes: Hidden layer 1 (5/10/20 Units) and Hidden Layer 2 (5/10/50) Units were tested to check the model performance
Different trials for hidden unit 1

No of Units	Train Loss	Test Loss
5	2098.5957	2355.7607
10	235.03624	242.80444
20	60.647392	123.99119

As the loss increases from unit size 20, the final units were considered to be 10

Different trials for hidden unit 2

No of Units	Train Loss	Test Loss
5	1175.067	750.74528
15	180.125	132.4917
25	80.125	123.99119

Hidden Layer 2 with 15 units are found to be optimal

10.3 Optimization Algorithms

Optimizers like Adam, RMS Prop, Adagrad and SGD with momentum was tried and the top resuls are shown as follows,

Optimizer	Train Loss	Test Loss
Adam	180.125	132.4917
AdaGrad	3456.25	4563.24
RMS Prop	234.56	345.23

Adam is selected among all the other optimizers

11 Improvement - LSTM Method

11.1 Hidden Unit size

Analyzing different unit sizes for the num_unit paramter for the basic RNN module,

Hidden units	Train Loss	Test Loss
1	25378.0	47399.0
5	168.23	245.34
10	55.277	124.22
50	15.13	245.46

Hidden unit of size 10 had the best result performance on test data.

12 Conclusion

Analyzing two different methods using Multi-Layered Perception (MLP) and LSTM using group Lasso technique, it seen that each of them have pros and cons. While MLP is simple and more interpretable, it needs the number of lag variables pre-specified. The LSTM, although a bit complex learns the number of lag variables required for the final prediction once it is provided with K lag values.

Thus, we would like to conclude saying that building the right model by selecting the best hyper parameters helped in identifying Granger causal relationship even in a constantly varying dataset like stock prices. This implementation could be used in various other domains having more predictable outcomes to identify such relationship with a very high accuracy.

References

[1] Alex Tank, Ian Covert, Ali Shojaie, Emily B. Fox. 2018. Neural Granger Causality for Nonlinear Time Series 1-16