A deep learning approach to financial time series: GRU embedded RNNs

Research Proposal

Spencer Frebel

Stockholm Business School
Advanced Financial Theory 7.5 HE Credits
Subject: Asset Pricing
Program: Master's Programme in Banking and Finance 120 HE credits
Autumn semester 2018
Word count:5213

Stockholm Business School

Stockholm University

Table of Contents

1 Overview & Introduction

Interpretation and prediction of time series is a challenge which cannot be understated. Traditional time series analysis ordinarily involves the use of autoregressive models, with varying degrees of results. However, where these models usually falter is with trends and cyclical oscillations, which may occur over a long-time scale dependencies. Thus, the purpose of the research proposal is to investigate whether a deep learning artificial neural network (ANN), more precisely, a Recurrent Neural Network (RNN) with Gated Recurrent Unit (GRU) cells, can showcase (or capture) these long-term dependencies by correctly forecasting financial time series, by specifically comparing the results to an Autoregressive Integrated Moving Average (ARIMA). The experiment shall be conducted in such a way where there will be two types of models, a multi-variate regression time series model, and a multivariate time series binary classification model. The output of the regression model shall be a prediction for the following time step(s) and the output for the classifier will be either 0 or 1 for each time step prediction, with the interpretation of a projected an increase or decrease of the financial time series.

It is common knowledge that financial time series prediction is among the most challenging problems to solve due to the large amount of volatility and random noise. Time series are often non-linear and highly time dependent. Trends can increase capriciously and decrease gradually, thus it is often not very likely to find financial time series which perfectly reverts to an average or a moving average, this poses a problem for traditional models, such as a univariate & multivariate ARIMA models, Simple Exponential Smoothing models, and Moving Averages (MA) models which rely on such parameters.

Financial time series forecasting has largely remained unchanged since the generalization of the Box-Jenkins method which was published in 1970. The method is archaic; however, it is still used and taught in modern day. There is some literature regarding the implementation of deep learning algorithms for time series prediction, however, since this is a relatively new field of study, it lacks a clear consensus of which algorithms are best suited for the task.

Various machine learning methods are now researched and implemented within finance, particularly deep learning, (generally described as more complex models when compared to shallow learning) have introduced new methods regarding forecasting problems where relationships between independent and dependent variables are in deep and layered hierarchy. Among the deep learning algorithms in research and implementation are RNNs, Feed Forward Neural Networks (FFNN), Convolutional Neural Networks (a type of FFNN) (CNN), Gated Recurrent Units (GRU), Long Short Term Memory (LSTM) and various amalgamations of models, again, with no clear consensus.

Deep learning methods can identify non-linear relationships within time series forecasting (Namin and Namin, 2018). Thus, it is an important research question, can the accuracy of prediction within deep learning algorithms outperform the ARIMA, MA, SES. To the best of my knowledge, there seems to be a lack of empirical evidence utilizing a RNN-GRU against the more traditional econometric forecasting techniques, such as the Box-Jenkins approach, ARIMA, MA, and SES. I propose the use of Gated

Recurrent Units within a RNN primarily due to the lack of literature for sequential modeling, (GRUs were only first proposed in 2014) as well as the potential for use in sequential modeling which was shown by Cho et al. (2014).

This research proposal aims to compare the ARIMA model and a RNN-GRU with respect to minimizing error rates or minimizing the total "cost" of the model of financial time series, namely equities/derivatives. This can be done either by taking the Mean Squared Error (MSE) or the Root Mean Squared Error (RMSE), since I am concerned with the magnitude of error of the prediction, I will thus use RMSE for the evaluation of both models, which by nature is sensitive to outliers. Furthermore, I intend to produce a RNN-GRU binary classification model and evaluate the model based on an F1 score.

It is important to clearly distinguish the terms "hyperparameter(s)" and "parameter(s)" as they are not synonymous, hyperparameter refers to parameters which can only be modified by a person, i.e. model structure and model selection. Parameters are what is to be interpreted and "learned/fitted" by the model. For the purposes of the proposal, I will only briefly describe the deep learning model as I plan on going much further in-depth in subsequent submissions to fully describe the model and the theory which motivates it.

2 Literature Review

Due to the nature of how new the entire field of study deep learning, with respect to finance, the resources are limited. However, in the last few years or so there is seemingly a surge of new literature regarding financial time series predictions using some sort of deep learning frameworks, i.e. a neural network. However, there seemed to be a dearth of empirical studies with RNN-GRUs with respect to financial time series prediction.

2.1 Empirical Asset Pricing via Machine Learning

This paper was the most comprehensive piece of literature published yet in the context of applying machine learning to asset pricing. However, the methodology used is somewhat similar to what I propose, with the caveat of lacking any clear direction pertaining to deep learning. The aim was not so much focused on price prediction but rather methods used to best identify risk premiums within stock prices via machine learning.

Furthermore, it aims to analyze two major research questions with respect to using machine learning to price assets, first, to understand the difference between in expected return across assets. And second, focus on the dynamics of the market equity risk premium. It states that the measurement of an assets risk is fundamentally a problem with prediction. The underlying problem of failure to correctly predict price is the nature of risk, however, this paper aims to answer the higher extrapolated cause rather than simply addressing the root cause of poor price prediction. The paper also states that the conditioning variables that account for the price of a stock has been shown to be in the hundreds (Gu et al., 2018), if not thousands. The paper does however mention that many of the stock prediction variables are highly correlated to each other. Therefore, if one were able to select proper conditional variables, machine learning would be optimal due to the ability to detect patterns in historical data coupled with the drastic reduction of degrees of freedom.

The paper has a two-method style of implementation, first, with different classification algorithms, where the outcome of the model is binary variable. Regression algorithms are utilized through means of “shallow” and “deep” learning. The article uses a rather common tactic of testing the success of such algorithms specifically, testing the MSE, as well as different techniques to avoid the overfitting (a phenomenon in which a model/algorithm is unable to generalize to other datasets) data such as hyperparameter tuning, parsing data correctly into training, validation, and test sets as well as other techniques which use “robustification” against outliers (Gu et al., 2018).

The authors began the analysis by first describing the common OLS of a linear regression model, while this model has been used in academia for a large portion of time, it does not allow for nonlinear affects which a sizeable problem is considering most asset pricing schemes follow a nonlinear path. The authors utilized the pooled OLS as a baseline. Following the baseline model, the authors implement a little more complex boosted regression and regression forest approach to pricing assets. Boosting is when one recursively combines prior forecasts from simplified decision trees to stack them on top of each other, more simply one must take prior binary forests over the course of the prediction and stack them in such a way where one binary forest is almost useless by itself, but together it creates an insightful prediction.

Another strong point that the article points out is the fact that theoretical or “traditional” literature has given little in the way of selecting conditional variables. So many factors must be correct in order for the data to be considered suitable the model. As with most financial data, equites/derivatives/most financial assets are nonlinear which calls into question the validity of older models which, in times make brash assumptions of linearity. The article goes on to state that machine learning is in fact explicitly created to approximate complex and nonlinear associations (Gu et al., 2018). Deep learning is particularly well suited for many prediction tasks which call for nonlinearity. The final point which is made in the article is parameter penalization and a conservative model selection complement each other in such a way as to avoid overfitting bias and false discovery (Gu et al., 2018).

2.2 A deep learning approach for credit scoring using credit default swaps

This paper aims to outline a deep learning technique to assess the credit rating of credit default swaps. Following the 2007-2008 global financial crisis, it became very evident credit scoring was, and still is, a key role in the management of risk. Deep belief networks with Restricted Boltzmann Machines were evaluated and compared with logistic regression, multi-layer perceptrons (MLPs), and support vector machines (SVMs). Performance was assessed via classification accuracy (Cuicui et at., 2017).

The dataset contained information on 661 companies and each of the models were given eleven input attributes with an outcome of three classification categories, i.e. rating classes of A, B, and C, with A receiving the highest credit score. The comparison metric used a 10-fold cross-validation. The results were very compelling, a Multinomial Logistic Regression scored the worst in terms of accuracy, with just over 77% accuracy and the deep belief network with Restricted Boltzmann scored the highest of 100% for the classification of the ratings with zero False Negatives and zero False Positives.

The paper is relative to the presented research proposal in terms of the prediction of risk in the broadest scope, a prediction of financial time series and the classification of credit ratings both fall under risk management. Pricing discrepancies would not exist if the risk of any asset was completely transparent. The most striking limitations are the type of data, structure of the data, and structure of the deep learning models which were utilized. Additionally, this paper utilizes data which is non-temporal, i.e. non-sequential data thus further limiting the application towards my proposal.

2.3 A deep learning framework for financial time series using stacked autoencoders and long-short term memory

Each topic by itself has been studied in-depth, however, comparative studies have been done in the context of time series data, however, with respect to equity/derivative price prediction, there appears to be room for further investigation (Bao et al., 2017). Bao et al. (2017) lay down a very strong foundation for the literature regarding the topic. Furthermore, reviews were performed on various models regarding deep learning, particularly, ANNs. The authors examined a few different architectures for ANN's to accurately predict financial time series. They seemingly were able to flawlessly execute a very complex model via different machine learning architects. The model however had its problems, namely suffering from a very "computationally expensive" model which took an inordinate amount of time to properly fit ("train") the model. They conducted six stock indices to test the prediction ability of the model (Bao et al., 2017). Through the utilization of a "mix-and-match" style of selection, they showed the predictive power of the model of their most intricate model to be the most powerful at a 5% confidence level.

The conclusion reached by the paper was the ability to predict the following price with a significantly high degree of accuracy. Not only were they able to predict accuracy but also, perhaps more important, was the fact they were able to create a profitable model when compared with the industry benchmarks. The main drawbacks which were reflected upon in the paper, they proposed a more intensive method for selecting hyperparameters to optimize both the time and efficiency in terms of computational power. Hyperparameter optimization is still in early stages as this is a major hinderance for researchers/practitioners, as no definite relationship between hyperparameters and desired output has been defined.

This architecture of this algorithm is different, mainly via the use of autoencoders, which has a different input, hidden, and output layer architecture of model and the use of LSTMs rather than the GRUs I intend to implement. The differences are significant from the algorithm proposed as the number of variables used for input are greater and the structure of the algorithm is much different.

2.4 A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction

M Xie et al. (2002) published a paper which is a topical comparison between the Box-Jenkins ARIMA model for and a neural network. The paper aims in some sense to create some sort of benchmark of what should expected when a comparison is made. Unfortunately, the paper is dated in terms of the speed of change within technology. The paper, is brief and concludes that a fitted ARIMA model can perform at the same capacity as an ANN in the prediction of a time series object. Furthermore, the study used a FFNN, where each input is not assumed to be sequential of the prior, which was shown here to be less effective compared to the ARIMA model. If M Xie et al. (2002) were to create a FFNN in such a way that makes predictions whether the price of an asset increases or decreases for a given time step, results would be much different. The drawbacks here are the type of structure, the complexity of the deep learning model, and the use of additive variables. These are significant differences compared to this research proposal.

2.5 Neural Networks as a Decision Maker for Stock Trading: A Technical Analysis Approach

Thawornwong et al. (2003), implemented a unique approach to application of deep learning for the prediction of stock prices. With technical indicators as input data, rather than price time-steps, the authors attempted to create a meaningful prediction in the form of a classification algorithm. Technical indicators utilize historical data and volume movements for forecasting purposes. Their paper attempts to uncover the underlying nonlinear pattern of technical indicators. By utilizing three different neural networks the results show that the proportion of correct predictions and the profitability of trading strategies are significantly higher comparing to the benchmark or to prior studies (Thawornwong et al., 2003).

The obvious difference between my proposed approach and this paper is the use of FFNN vs. RNN. This seemingly insignificant difference creates a completely different structure of the model. The authors use a FFNN to capture the nonlinear long-term inter dependencies rather than a RNN.

2.6 Stock Market Value Prediction Using Neural Networks

Naeini et al. (2010) created a two model study which used a feed forward multi-layer perceptron and a RNN in order to predict the equity price of a company which utilized only prior share information. The authors had rather some interesting findings, their study indicated that the MLP was better at predicting stock value changes rather than the RNN. The authors noted that the error of the prediction of the amount of value was lower with the MLP which led them to conclude that the MLP was better than changes in price. Typically, these findings would indicate a contradiction to my research proposal in so far as the structure of the deep learning model, however, I could not find in the paper the specific hyperparameters which were used on the model. Again, no definite empirical relationship between hyperparameters and accuracy has been discovered, however, with no mention of the hyperparameters the findings cannot be validated and the replicability of the findings would be difficult.

2.7 Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling

Chung et al. (2014) published a paper to compare the different types of recurrent units with in an RNN. Their evaluation focused on the LSTM, GRU, and the “vanilla” hyperbolic tangent (tanh) of the prediction of sequential data, although not specifically financial time series. Their findings demonstrated the superiority of both the gated units over the tanh unit. Furthermore, their findings also found that GRUs were superior to LSTMs when the dataset were small (Chung et al., 2014). A possible reason for this is that GRUs have fewer parameters.

The LSTM is the primary ‘mainstream’ variation of the vanilla RNN for sequential modeling as it has demonstrated the ability to interpret long term inter dependencies, while solving the exploding/vanishing gradient problem (discussed in the methodology). However, the proposed GRUs have the ability also be suitable for RNNs which use sequential data (Chung et al., 2014).

3 Methodology & Data

This section describes the testing setup environment and the methodology, implementation, and evaluation for the different models. The order shall begin with the description, partitioning, construction, and implementation of the RNN-GRU model (regression and classification) and the methods used for data normalization & preprocessing. Next, I’ll describe the process I intend to use to find the optimal hyperparameters for each model as well as model evaluation. Finally, I will describe the process for generating and initializing an ARIMA model for comparison purposes.

3.1 Recurrent Neural Network

The motivation to use RNNs is not obvious, the main reason feedforward neural networks are viewed as inferior is not treating each input as dependent on the prior information, non-sequential. For instances such as image recognition, the input data is not dependent, i.e. if the model predicted a car in one instance that does not necessarily mean the next instance will anything related to a car, thus the motivation for an augmented feedforward neural network is motivated by way of sequential data. The augmentation is made in way of recursion. The RNNs are connectionist models which can capture the dynamics of sequences via cycles in each of the nodes. Unlike a more standard feedforward neural network, recurrent neural networks retain a state which can represent information form an arbitrarily defined context window. Historically, RNNs have been very difficult to train due to often millions of parameters, recent advances in network architectures, optimization techniques, and parallel computation have largely enabled large-scale learning to occur. In the recent years long short term memory (LSTM) and other architectures have shown powerful potential when it comes to sequential learning (Lipton et al., 2015).

RNNs are augmented feedforward neural networks which have been modified by the inclusion of edges which span across the different time steps, thus introduce a time component to the model. At time t, nodes with recurrent edges receive input from the current data x^t and from the hidden node at values h^{t-1}, which was the value in the previous state of the model. The output prediction y^t if calculated using given the hidden node values h^t at time t. The input x^{t-1} at time $t - 1$ can influence not only the output prediction y^t but also from the recurrent connections in the prior time steps (Lipton et al., 2015).

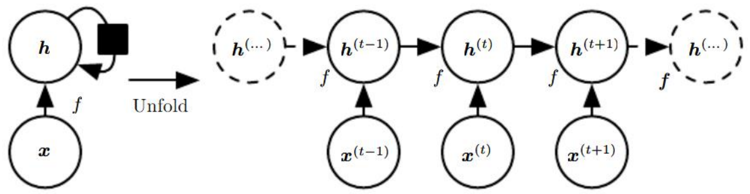
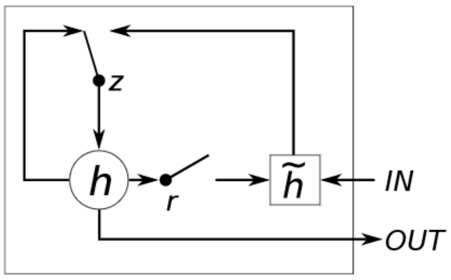


Figure 1: A simple visual depiction of a unfolded computational graph of an RNN with no outputs, information is simply processed from the input $\boldsymbol{x}$ into the current state of the model $\boldsymbol{h}$ which is subsequently passed through time, the left depicts a circuit diagram, while the black square shows a delay of a single time step, on the right, the RNN is unfolded where each node is associated with one time instance. After the input is passed to each cell through time an activation function f is used and subsequently passed to the next cell. Source: (https://www.deeplearningbook.org/contents/rnn.html)

3.2 Gated Recurrent Unit

Bengio et al. (1994) observed it can be difficult to train RNNs to capture long term dependencies because the gradients tend to either disappear or become so large that the activation function in each recursive step is useless. Due to the nature of activation functions, stacked with the recurrent nature of the model, each time input is passed through the node, the value of the input becomes increasingly closer to one (exploding gradient) or closer to zero (vanishing gradient). This makes gradient based optimization (the most popular method in which the multivariate gradient (the connection between each nodes) is minimized via the chain rule in calculus) very difficult as the variations in each of the gradients has different magnitudes. Thus, the solution becomes to either modify the way in which a model is trained for example, using a gradient clipping method which runs the risk of removing data which is relevant to the prediction or use a second-order method, which is probably to be less sensitive to the issue if the second derivatives follow the same growth pattern as the first derivatives (Chung et al., 2014).

The approach is to implement a more sophisticated activation function. The first successful implementation was called a long short-term memory (LSTM) cell which was synthesized in 1997 by Hochreiter and Schmidhuber. More recently, the gated recurrent unit proposed by Cho et al. (2014) is the focus of the proposal. RNNs which have either LSTMs or GRUs have performed well in capturing long-term dependencies.



(b) Gated Recurrent Unit

Figure 2: r and z are the reset and update gates, and h and $\tilde{h}$ are the activation and candidate activation. Source: (https://arxiv.org/pdf/1412.3555.pdf.)

The activation h_t of the GRU at time t is a linear interpolation between the prior activation h_{t-1} and the candidate activation $\tilde{h}$ such that:

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t$$

in which an update gated z_t determines the how much the unit will update the activation. The update gate is computed as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

where σ is the logistic sigmoid function such that $\sigma(x) = \frac{e^x}{1+e^x}$

The candidate activation is computed as:

$$\tilde{h}_t = \tanh(Wx_t + U(r_t h_{t-1}))$$

Where $tanh$ is the hyperbolic tangent function such that $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

When r_t is close to 0, the reset gate effectively makes the unit read as if it was the first symbol in the sequence, which allows it to forget the data from the prior unit.

The reset gate r_t is computed as:

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

(Chung et al., 2014)

3.3 Data Extraction, Preprocessing, and Software Packages

Now that the model has been described (briefly), I can now describe the process regarding generating and processing data. The first step in the process is obtain suitable data for which I can analyze. Extracting the data from Bloomberg is preferable as I will have the most up to date and accurate data. The format of the data is not as important although, it would be preferable to have the data in a .csv format. Next, I will partition the data into three different sets for the RNN-GRU model, training, validation, and testing. The exact ratio of each has yet to be determined, however, many models use a 70% training/validation and 30% for testing.

3.4 Window Partition

This first step partitions the data into discernible units known as “windows” which can be thought of as small pieces of the larger data set that are next to each step in the time series, i.e. transforming a dataset from its original form into sequential data for which the model can be trained on and made to make predictions for the following time steps, consider an example where the price of an asset formatted as such where each price corresponds to a date:

Date (d)	**Price (p)**
d1	p1
d2	p2
d3	p3

The data above needs to be reshaped so it can be used as a sequential problem. Once the data is transformed, it changes to a supervised regression problem where the order of the data is upheld. X represents a matrix and y represents the target variable as a column vector. N/A are values which are not relevant/known at the time of input (Li, 2018).

X	**y**
N/A	p1
p1	p2
p2	p3
p3	N/A

This method is known as the sliding window method, which follows a Multi-Input Multi-Output strategy of forecasting which models a multiple input and output mapping, while still preserving the stochastic dependencies between the predicted values. While it is possible for RNN to be given one input at a time, the internal state of the network will need to memorize all relevant info. Using this method allows the model to operate directly with lagged values as input (Bandara et al., 2018)(Li, 2018).

More formally, this can be expressed as Z_0 up to Z_t with each window containing price p_i with size w

$$Z_0 = (p_0, p_1, \dots, p_{w-1})$$

$$Z_t = \left(p_{tw}, p_{tw+1}, \dots, p_{((t+1)w-1)}\right)$$

To predict the next window Z_{t+1}

With the goal of a regression function be a function of the aggregate windows:

$f(Z_0, Z_1, \dots, Z_t) = Z_{t+1}$

(Li, 2018)

3.5 Data Scaling & Normalization

Due to the nature of equity/derivatives prices having an absolute value increase over time. This reality of equity prices creates the problem that most values in the testing set are out of scale, which indicates that the resulting regression model must learn values it has not seen before. Thus, it is necessary to normalize the price. For the regression task, I simply take the daily return where all values in each window are divided by the prior daily return window Z_{t-1}. Note, I intend to normalize all of the data, even the test set, I am aware this creates a bias assumption of mean reversion of data.

For the binary classifier model, I am only interested in the direction and subsequently only interested if the testing data from each window is positive or negative, thus I am only interested in the sign of each of the column vector values (Li, 2018).

3.6 Hyperparameter Selection & Optimization

When training a machine learning model, the models have input parameters, intuitively these can be thought of as the "dials" and "cranks" of a machine that need to be manually modified, which is usually referred to as hyperparameters. Unfortunately, tuning these parameters for optimal performance is very costly both in terms of time but also computing resources.

An exhaustive grid search (EGS) is a common method which is utilized and can be thought of as brute force style of searching through a specified subset, more simply, given a range of each of the hyperparameters the grid search attempt each possibility. An EGS is typically guided by some performance metric, possibly a cross-validation of the training set (Chin-Wei et. al, 2010). GRUs have various hyperparameters, thus, I will forgo an EGS and use a Bayesian global optimization method, a method which uses Bayesian inference and a Gaussian process to optimize the hyperparameters of function that is not known using a validation set. An epoch refers to the number of times the input training data X is passed through the model to the output cell y. The available hyperparameters are as follows (Bandara et al., 2018):

Parameter	**Min Value**	**Max Value**
Window Size	1	20
Learning rate	0.1	0.0001
Hidden Layers	2	8
Num of GRUs	4	256

Num of Epoch	100	1000

3.7 Model Evaluation RNN-GRU

For the multivariate regression model for the RNN-GRU, intend to measure the results with the Root-Mean-Square Error (RMSE). This metric is commonly used for assessing the accuracy of model. It measures the aggregated absolute value of differences between the model prediction and the correct values. The most obvious benefit is the penalization of large errors within the prediction set while keeping the scaling of the scores consistent.

$$RMSE = \sqrt{\left(\frac{1}{N}\right)\sum_{i=1}^{N}\left(x_i - x_p\right)^2}$$

Where N is total observations, x_i is value and x_p is the projected value.

For the multivariate binary classification model, I intend to measure the success of the model via an F-measure. Which considers both the Precision and Recall of any given test. Precision measures the ratio between the correct number of predictions and the total predictions. The Recall measures the ratio between the correct number of predictions and the total number of the given direction.

$$Precision = \frac{Acutal\ Positives}{Acual\ Positives + False\ Positives}$$

$$Recall = \frac{Actual\ Positives}{Actual\ Positives + False\ Negatives}$$

$$F_1 score \ = 2 \ * \left(\frac{Precsion * Recall}{Precision + Recall}\right)$$

3.8 Autoregressive Integrated Moving Average (ARIMA) Model

ARIMA is a generalized model of the Autoregressive Moving Average (ARMA) which combines an AR process with an MA process to create a composite model of a time series (Pesaran, 2015). The ARIMA model uses the acronym ARIMA (p, d, q) such that:

AR: A model which utilizes the dependencies between an observation and number of lagged observations(p).

I: In order to create a stationary time series we must measure the differences of observations at different time (d).

MA: Considers the dependency between the observations and the residual error term when a moving average is used to the lagged observations (q).

Where p is the lag order, d is the degree of differencing and, q is the order of the MA (Pesaran, 2015).

The ARIMA is a type of model which attempts to capture the temporal dependent relationships within time series data. ARIMA is a linear regression based model, which implies that is best for the forecast to be one-step beyond the sample. For my model comparison, I intend to create a forecast that is multiple steps out the sample. To do this I need to ensure that each time step the model is re-fitted to create the best estimation for the subsequent step (Brownlee, 2017).

3.9 Model Set-up

To stay consistent with the RNN-GRU model, I will use the same training/validation to test split of 70% train/validate ("in-sample") and 30% ("out of sample) as the sample above. Once the data is split into two sets, I will create two data structures to hold each iteration of the training data as labeled "history", and the predicted values for the test as "prediction" (Namin and Namin , 2018).

I intend to establish a baseline model for the ARIMA of (6,1,0) where the lag value is set to 6 for the AR, a difference order of 1 to ensure that the time series is stationary and finally 0 where the window size is not important (Namin and Namin , 2018). This may not be a perfect estimation; however, it is generally a solid baseline. I intend to optimize this number by performing an EGS, which is simpler than the EGS which would be needed by the RNN-GRU. Only three parameters require estimation by iterative trial and error and generally the d value never goes above 2 and for a forecast the value for q shall also be close to 0. Similar to the RNN-GRU model, I intend to use a Bayesian optimization method to tune the hyperparameters my multi-step projection and have an optimized ARIMA model I can make my predictions.

3.10 Model Evaluation ARIMA

I intend to evaluate the ARIMA by measuring the RMSE of the projected values and compare the results with the multivariate RNN-GRU regression model. I intend to evaluate a binary classification tool with an F1 score.

4 Limitations

The RNN-GRU model in specific has obvious limitations, the time is takes to train a model to capture long term dependencies can take many hours if not weeks. This would not be a practical implementation into any sort of high frequency trading strategies which have become increasingly popular in modern times. Additionally, a very important distinction to make pertaining to both the regression model and the binary classifier model is are future values are predicted based on values which have already occurred, this model is more akin to a back-test. The deep learning models will not use the entire training data set to predict the next value.

References

Bandara, Kasun, et al. “Forecasting Across Time Series Databases Using Recurrent Neural Networks on Groups of Similar Series: A Clustering Approach.” *ArXiv.org*, 2018, arxiv.org/pdf/1710.03222.pdf.

Bao et al. “A deep learning framework for financial time series using stacked autoencoders and long-short term memory.” (2017) PLoS ONE 12(7): e0180944. https://doi.org/10.1371/journal.pone.0180944)

Brownlee, Jason. “How to Create an ARIMA Model for Time Series Forecasting in Python.” Machine Learning Mastery, 1 Aug. 2018, machinelearningmastery.com/arima-for-time-series-forecasting-with-python/. (Brownlee, 2017)

Chung, Junyoung, et al. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.” ArXiv.org, 2014, arxiv.org/pdf/1412.3555.pdf.

Gu, Shihao and Kelly, Bryan T. and Xiu, Dacheng, “Empirical Asset Pricing via Machine Learning (2018). Chicago Booth Research Paper No. 18-040” University of Chicago http://dx.doi.org/10.2139/ssrn.3159577

Hsu, Chih-Wei, et al. “A Practical Guide to Support Vector Classification.” Department of Computer Science, National Taiwan University, 2010, www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

Li, Edwin. “LSTM Neural Network Models for Market Movement Prediction.” Kth.diva-Portal.org, Kungsliga Tekniska Högskolan, 2018, kth.diva-portal.org/smash/get/diva2:1231223/FULLTEXT02.pdf.

Lipton, Zachary, et al. “A Critical Review of Recurrent Neural Networks for Sequence Learning.” ArXiv.org, 2015, arxiv.org/pdf/1506.00019.pdf.

Luo, Cuicui, et al. “A Deep Learning Approach for Credit Scoring Using Credit Default Swaps.” Science Direct, Engineering Applications of Artificial Intelligence, 2017, doi.org/10.1016/j.engappai.2016.12.002.

Naeini, Mahdi, et al. “Stock Market Value Prediction Using Neural Networks.” 2010 International Conference on Computer Information Systems and Industrial Management Applications, 2010, people.cs.pitt.edu/~pakdaman/papers/CISIM2010.pdf.

Namin, Sima, and Akbar Namin. “Forecasting Economic and Financial Timer Series: ARIMA vs. LSTM.” ArXiv.org, Texas Tech University, 2018, arxiv.org/ftp/arxiv/papers/1803/1803.06386.pdf. (Namin and Namin , 2018)

Pesaran, M. Hashem. Time Series and Panel Data Econometrics. Oxford University Press, 2016.

Roondiwala, Murtaza & Patel, Harshal & Varma, Shraddha. (2017). Predicting Stock Prices Using LSTM. International Journal of Science and Research (IJSR). 6. 10.21275/ART20172755.

S.L Ho, et al. “A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction.” Computers & Industrial Engineering, Volume 42, Issues 2–4, 2002, Pages 371-375, ISSN 0360-8352, https://doi.org/10.1016/S0360-8352(02)00036-0.

Suraphan Thawornwong, et al. “Neural Networks as a Decision Maker for Stock Trading: A Technical Analysis Approach.” International Journal of Smart Engineering System Design, 5:313–325, 2003.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994.

Stockholm Business School

Stockholm University
SE-106 91 Stockholm
Tel: 08 – 16 20 00
www.sbs.su.se

Stockholm University