Algorithmic Trading using LSTM-Models for Intraday Stock Predictions

David Benjamin Lim
Department of Mathemtics
Stanford University
benlim@stanford.edu

Justin Lundgren
MS&E
Stanford University
julu1@stanford.edu

Abstract

We investigate deep learning methods for return predictions on a portfolio of stocks in the information technology sector. We deploy standard time series models alongside with an LSTM network and recently developed method called R2N2, which is an LSTM network applied to the residuals from the standard time series models. We obtain very modest results in terms of standard metrics such as accuracy, mean squared error and AUC. However, in terms of financial metrics, we find that our model allows us to performs better than what the market portfolio does. Applying a simple short/long strategy we obtain a daily return of 0.49% using the LSTM.

1. Introduction

The past decade or so has shown that deep learning is an extremely powerful tool in machine learning, achieving success in areas such as sequence learning, speech recognition and image classification [2]. Crucial to the success of deep learning in these areas is a philosophy advocated for in [5], similar in spirit to the seminal essay of Wigner. Namely, in order to learn features for the task at hand, a simple model fed with a lot of data is better than a more complicated model fed with less data.

Motivated by the above, and in particular the results of [3] and [4], the goal of this project is to apply the tools of deep learning to the task of predicting stock returns. The methods we will use are a VAR/VARMAX as a baseline, an LSTM model and finally an R2N2 model. However, we note that the task at hand is very challenging for the following reasons. First, stock data has an extremely high noise to signal ratio. Second, due to the competitive, profit driven nature of finance, it is in the best interests of professional financiers that the amount of *publicly* available data on standard finance APIs (Alpha Vantage, Yahoo Finance, etc) be *limited*.

This article is organized as follows. In Section 2, we describe the dataset of our project. We begin Section 3 by describing a preprocessing step done on the raw data. We then give a brief explanation of how our models work. Section 4 contains the results of our models on the test set, a simple trading strategy and an evaluation of this strategy on the data. We conclude in Section 5 with a discussion of future work.

2. Data

We use a data set available online [1] that has intraday time series data at one minute intervals for all stocks in the S&P 500 between 9/11/2017 and 2/16/2018. However, in order to have a feasible strategy to act on, we only use timestamps that are five minutes apart. Each timestamp reports close price, high price, low price, open price and volume for 502 stocks. A caveat about the dataset is that any stock that entered or exited the index in this time frame is omitted from the data set.

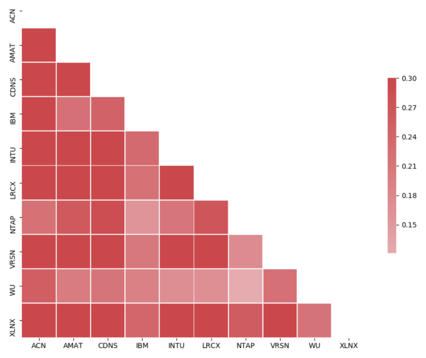


Figure 1. Correlation plots

We run our project on 10 stocks from this data set, which we choose as follows. First, we partition the stocks in the S&P 500 into sectors, e.g. finance, information technology and energy. The reason being that stocks in the same sector should be better correlated as opposed to stocks in different sectors. We posit that the more correlated the data is, the larger the signal to noise ratio will be, thus enabling our models to produce beter stock predictions. Therefore, our initial data analysis was to find a portfolio of stocks that were highly correlated. We ran pairwise correlations among the sectors and identified the information technology sector as a sector where it would be able to easily pick out a portfolio of correlated stock. At the end of this analysis we settled on the stocks ACN, AMAT, CDNS, IBM, INTU, LRCX, NTAP, VRSN, WU, XLNX. The correlations between these stocks are presented in Figure 1 above.

Finally, we split our dataset into the standard partition of 70/15/15 for training/validation/testing.

3. Method

In this section, we recall the different types of models that we use. The first of these is our baseline.

3.1. Feature Extraction

The basic time series data usually requires some preprocessing before it can be used. Following [10], we apply the min-max-scalar

$$\tilde{x} = \frac{x - \min(x)}{\max(x) - \min(x)}.$$

to the features that clearly not are stationary. In particular, we apply this to the volume an the open price of each time period. For the high and low prices, we use the simple difference between high and low price to get a measure that is close to zero and that contains information about the current fluctuation in the stock. In total our features are the difference between high and low, min-max-scaled volume and open price and previous returns.

Following [7] and [10], we also experiment with applying sine-transformations and Laguerre polynomials as part of the feature engineering. This was omitted in the final results however, as the results on the validation set did not increase with these additional features.

3.2. VAR and VARMAX

Let y_t be a k-dimensional discrete-time stochastic process. A vector autoregression model of p describes a linear relationship between y_t and the previous p time steps $y_{t-1}, \dots, y_{t-p}$. More precisely, we model

$$y_t = \sum_{i=1}^{p} A_i y_{t-i} + A_0 + \varepsilon_t.$$

Here $A_0, \dots, A_p$ are parameters to be determined and ε_t is white noise satisfying $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$. In our setting, k will be the number of stocks (so $k = 10$) and y_t will be the returns (closing minus opening price) of our stocks. We choose to model a "first order difference" rather than on the nose prices as these rarely satisfy the assumption of zero mean.

It is common in financial modelling to consider the case when the endogeneous variables y_t depend in addition on exogeneous variables x_t and a moving average of the white noise. This is called a vector autoregressive (VAR) moving average (MA) with exogeneoux variables (X) of order (p, q), VARMAX. It models

$$y_t = \sum_{i=1}^{p} A_i y_{t-i} + A_0 + Bx_t + \sum_{j=1}^{q} B_j \varepsilon_{t-j} + B_0.$$

Our choice for exogeneous variables x_t are the features as defined in the previous subsection.

Finally, the hyperparameters p and (p, q) in the VAR and VARMAX are tuned via a basic grid search. Our criterion for optimal hyperparameter is the one that outputs the smallest MSE error on the validation set within a predetermined range. Our grid search produces $p = 1$ and $(p, q) = (1, 2)$. This is consistent with the fact that information technology stocks are rather volatile and do not have any apparent seasonality.

3.3. LSTM

Long short-term memory networks were first introduced in [6] and have been used successfully in image and text classification tasks. At present, the exact structure of an LSTM layer is too long to describe and we refer the reader to [9] for details. Nonetheless, the key idea is that each layer consists of LSTM units, each of which has the ability to manage memory via a forget, input and output gate.

3.3.1 Implementation of LSTM

First, we transform our data into an array of shape $(10, n - p, p, g)$. Here n is the number of training samples, p is the lag order and g refers to the number of exogenous features. From section 3.1 we have that $g = 4$. Table 1 shows the pipeline of the network.

Layer	Units/Rate
LSTMx10	4
Dropout	0.1
Concatenate	-
LSTM	10
Dropout	0.1
Dense	10

Table 1. LSTM network

The first LSTM-layer is an individual layer for all ten stocks and the second LSTM-layer is a joint layer constituting the outputs from the previous layer. The rationale for this structure is that the first layer should remember individual dependencies and the second layer should be able to learn dependencies between returns of different stocks. Moreover, we apply a dropout with rate 0.10 to decrease the risk of overfitting. Finally, a fully connected dense layer is applied. For training, we use the customize mean squared error

$$\ell(\texttt{data}) = \frac{\sum_{i=1}^{n}(r_i - \hat{r}_i)^2}{\sum_{i=1}^{n}\mathbb{I}\{\text{sign}(r_i) = \text{sign}(\hat{r}_i)\} + \epsilon} \tag{1}$$

where $\epsilon = 0.01$ in our case, that penalizes extra for getting the sign wrong.

Figure 2 shows the hyper parameter tuning for this model. Due to computational limitations, we only tune the most important hyperparameters, namely the lag order p, the learning rate and the amount of units in the first LSTM-layer. We start with a reasonable *ansatz*, and then sample 20 values randomly within a range of the *ansatz*. The results of this random search are shown in the figure below. We get 4 LSTM units, a lag order of 24 and a learning rate of 10^{-4}.

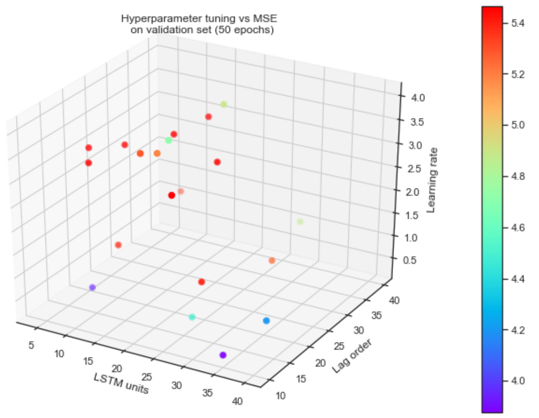


Figure 2. Hyperparameter tuning. Note that learning rate and MSE are printed on a negative log-scale.

3.4. R2N2

The R2N2 network was introduced by [4], who apply it using an RNN network. The idea is that some standard time series model $f(\cdot)$ (e.g. VARMAX) should be able to capture linear dependencies in the data and that the network $h(\cdot)$ should be able to map the non-linear dependencies. Thus we predict $\hat{r}_i = f(r_{i-1}, x_{i-i})$ where x denotes the exogenous features. Once we have $\hat{r}_i$ we obtain the residual $e_i = r_i - \hat{r}_i$ and predict $\hat{r}_{i+1} = h(r_i, e_i)$. Moreover, we implemented R2N2 using the exact same structure as the LSTM and the residuals from the VARMAX(1,2)-model.

4. Results

First we compare the proposed models with each other[1], then we go into detail of the main model and finally we investigate how this model can be utilized as a robust trading strategy.

4.1. Model evaluation

Despite being trained on the customized loss function defined in (1), the models are evaluated on the basis of several metrics. Table 2 shows the three proposed models and their scores for the mean squared error (MSE), the accuracy in terms of predicted positive/negatives returns, the return based on the unbiased threshold $t = 0$ and the Sharpe Ratio (SR). The MSE does not really give uany valuable information on how to utilize the model, but as the model is trained to minimize the customized MSE in (1), we still present this metric.

Moreover, simple 0-1 accuracy (Acc) is neither ideal since it does not contain any information about the magnitude of the predicted return. However, since we are dealing with such a short time frame, returns are regularly on a very small magnitude and therefore 0-1 accuracy can actually entail some level of security when investing. Moreover, the return is based on the simple strategy of going long when the model predicts a positive and going short when the model predict a negative[2]. Finally, the Sharpe ratio

$$\text{SR} = \frac{\mathbb{E}[r] - r_f}{\sqrt{\text{Var}(r)}}$$

is a metric of return versus risk, that is calculated using sample mean and sample standard deviation. r_f denotes the risk-free rate, which it is assumed to be zero, since we are dealing with such a small time interval.

Model	MSE	Acc	Return	SR
VARMAX	1.21e-5	0.506	3.73	2.16e-2
LSTM	1.56e-5	0.516	8.18	6.84e-2
R2N2	8.84e-6	0.501	3.11	3.04e-2
Market	8.62e-6	0.484	−5.53	−2.02e-2

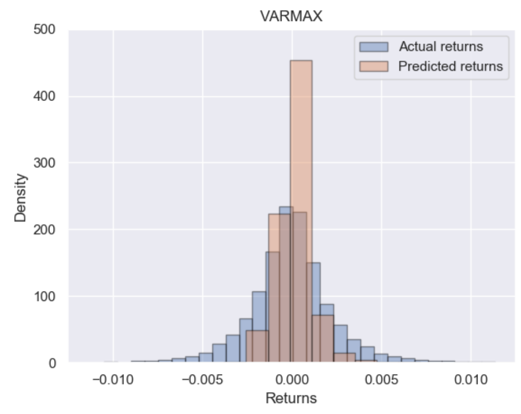
Table 2. Model metrics. Market corresponds to a constant long position where the MSE is evaluated on always predicting a return of 0.

As we see from table 2, the LSTM model scored best over all. Its returns correspond to a daily return of 0.49% since this test data is 16 trading days. In comparison with [3], this is along the lines of what we assume to achieve. Moreover, the market strategy is defined such that it entail information about the test data. The model based strategies all allow shorting, which explains why they performed so much better on the return and SR metric (note that just doing the opposite of the market strategy would have changed the

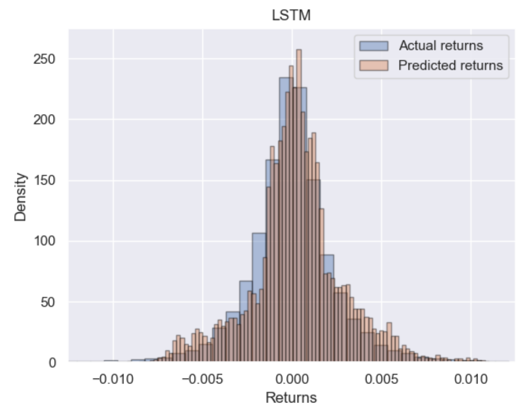
[1]Note that we do not include any figure of the predicted time series on an axis, as they all look very similar due to the short time interval.

[2]For the sake of simplicity, we assume zero transaction cost when shorting.

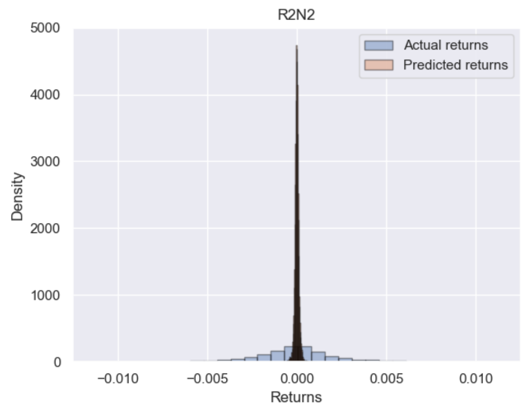
sign of the return and SR). Moreover, the market has the lowest MSE, which is expected since returns in such small time intervals are distributed very close to zero. To show how well these models map the distribution of the returns, we introduce the histograms in Figure 3.



(a) VARMAX



(b) LSTM



(c) R2N2

Figure 3. Histograms over the models. All actual bins are set at 200, whereas as all predicted bins are set at 100.

As figure 3 portrays, the overall best distribution comes from the LSTM model. It is however, surprising that the R2N2 predicts all returns so close to zero. Perhaps removing the ground features and adding the VARMAX residuals forces the model to predict values close to zero in order to obtain a low MSE.

4.2. Stock evaluation on the LSTM model

Table 3 shows the same metrics but for each individual stock.

Stock	MSE	Acc	Return	SR
ACN	1.470e-5	0.498	−1.168	−0.308e-2
AMAT	1.359e-5	0.538	38.16	7.275e-2
CDNS	1.991e-5	0.522	0.823	0.364e-2
IBM	1.487e-5	0.512	4.057	1.767e-2
INTU	5.208e-5	0.541	34.26	11.144e-2
LRCX	1.543e-5	0.460	−19.69	−4.800e-2
NTAP	2.432e-5	0.553	26.39	4.396e-2
VRSN	8.662e-5	0.521	11.37	3.590e-2
WU	1.089e-5	0.525	1.438	0.563e-2
XLNX	2.847e-5	0.496	−5.283	−1.490e-2

Table 3. Individual stock metrics using the trading strategy from the LSTM model.

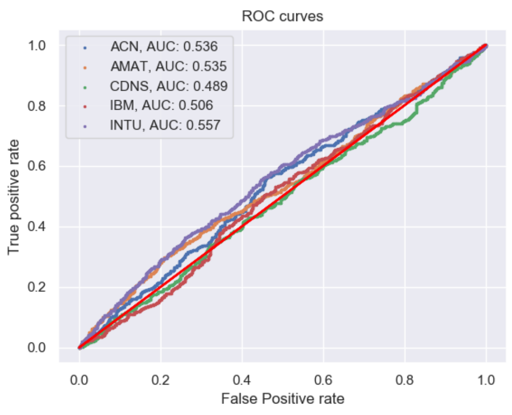
In order to benchmark the results in table 3 we also present the same market metrics, but for each individual stock in table 4

Stock	MSE	Acc	Return	SR
ACN	3.782e-6	0.497	−1.560	−0.542e-2
AMAT	1.304e-5	0.493	−10.59	−2.271e-2
CDNS	9.580e-6	0.465	−19.79	−5.488e-2
IBM	3.532e-6	0.493	−9.408	−4.065e-2
INTU	4.508-6	0.508	5.539	2.114e-2
LRCX	1.214e-5	0.477	−12.98	−2.979e-2
NTAP	1.979e-5	0.478	−4.568	−6.003e-2
VRSN	6.005e-6	0.515	8.589	2.782e-2
WU	6.867e-6	0.437	−4.337	−1.207e-2
XLNX	6.992-6	0.478	−5.290	−1.493e-2

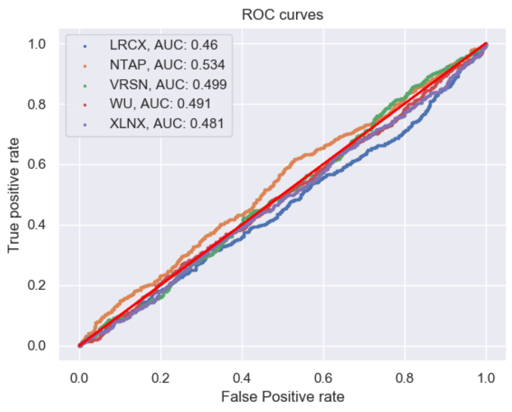
Table 4. Individual stock metrics using the market strategy.

From Table 4 it is evident that the combined market of these ten stocks was on a downwards trend trough out the time period of the test data. This means that allowing short positions in the trading strategy has clearly biased the overall evaluation of the method.

To further evaluate evaluate how well the model works on this portfolio of stocks, we present Figure 4, containing the ROC curves of the stocks in our portfolio. The ROC curves are created by looping the threshold t from the minimum predicted return to the maximum.



(a) ACN, AMAT, CDNS, IBM, INTU



(b) LRCX, NTAP, VRSN, WU, XLNX

Figure 4. Histograms over the models. All actual bins are set at 200, whereas as all predicted bins are set at 100.

From the combined results of the ROC-curves and the metrics in Tables 3 and 4 it is hard to say that the model has predictive power on the portfolio as a whole. Some of the AUC scores are surprisingly low whereas some are in the expected range for a task like this one. For example, the return strategy on the stock INTU was very fruitful, and it is also the stock with the highest AUC score. On the other hand, for the stock LRCX, the AUC is below 0.5 and the strategy return is also worse than that of the market strategy.

4.3. Trading strategy evaluation

Based on the joint validation of all three models, we chose the threshold for predicting positive/negative returns to be the obvious one, namely zero. The underlying reasoning for this decision is that all predicted returns are distributed close to zero, so tuning one "optimal threshold" on one model with the validation set, will most likely be uncorrelated with the optimal threshold for the test set. Therefore, we present figure 5 that shows how the Sharpe Ratio of the portfolio fluctuates as we change the threshold.

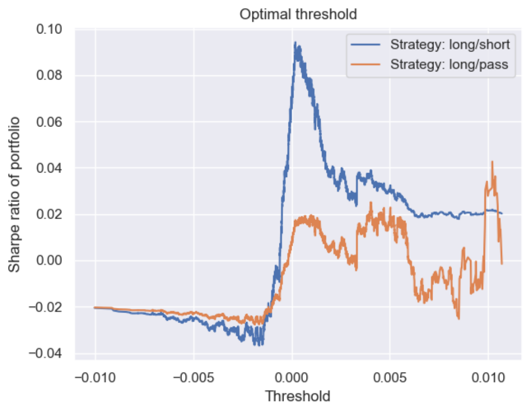


Figure 5. Sharpe ratio as a function of trading strategy on the test set.

As we see, our threshold was definitely now the optimal one. It would have been able to almost double the return if we had chosen a slightly higher threshold. Moreover, the long/short strategy almost always beats the long/pass strategy, which is a consequence of the downwards trend of the market through out this time period.

5. Conclusion and future work

The goal of this project was to construct a model that could be used for algorithmic trading on a portfolio of stocks. In general, the model does not score very well on standard metrics. This is most likely explained by the fact that returns in general are very hard to model and in particular, returns of such a small time interval as 5 minutes are all distributed very close to zero and thus not easily separated. This makes it hard to evaluate how well our model captures any trends in data. When utilizing the trading strategy though, the model outperforms the market in both returns and Sharpe Ratio. This signals that the model, on average, seems to be able to distinguish between negative and positive returns of high magnitudes.

Furthermore, due to computational constraints and time limitations, we only deploy the model on a portfolio of 10 stocks. For a true algorithmic trading approach, we should have used more stocks and perhaps a more reasoned trading strategy. Moreover, the R2N2 approach is not completely rigorous since we used the same structure as the one for the LSTM. A more thorough approach would retune hyperparameters for the R2N2 model. Indeed, other authors [4] have achieved nice returns with a R2N2-network. It would have been interesting to shed some more light on this method.

6. Contributions

David Benjamin Lim worked on setting up Google Cloud/GitHub, implemented VAR/VARMAX and tuned hyperparameters. Justin Lundgren implemented the LSTM/R2N2 , analyzed results and evaluated the trading strategy. Both team members contributed equally to the writing of the final report. Justin Lundgren prepared the poster.

7. Code

For our code, please refer to the github repository at [8]. The VAR/VARMAX models were trained using the StatsModels API. The LSTM and R2N2 models were trained in Keras with a TensorFlow backend. All computations were done in Google Cloud's Deep Learning VM with a Nvidia V100 GPU.

8. Acknowledgements

We thank Prof. Tengyu Ma and Prof. Christopher Ré for a very inspiring course. We thank Anand Avati, Gael Colas, Atharva Parulekar and Abraham Starosta for their help and advice regarding our project.

References

[1] S&P 500 index intraday data with 1min intervals. Available at https://www.kaggle.com/nickdl/snp-500-intraday-data.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35:1798–1828, 08 2013.

[3] T. Fischer and C. Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270, 12 2017.

[4] H. Goel, I. Melnyk, and A. Banerjee. R2N2: Residual recurrent neural networks for multivariate time series forecasting. *arXiv preprint*, 2017. Available at https://arxiv.org/abs/1709.03159".

[5] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24:8 – 12, 05 2009.

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[7] Y. Li, C. Szepesvri, and D. Schuurmans. Learning exercise policies for american options. *Journal of Machine Learning Research - Proceedings Track*, 5:352–359, 01 2009.

[8] D. B. Lim and J. Lundgren. cs229.project. Available at https://github.com/benjalim/cs229.project.

[9] C. Olah. Available at https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[10] G. Petnehazi. Recurrent neural networks for time series forecasting. *arXiv preprint*, 2019. Available at https://arxiv.org/pdf/1901.00069.pdf.