Jorge Alberto Rodriguez

Eduardo Aguilar

Kshitij Singla

Rachit Maheshwari

PROJECT SUMMARY

Ever since the advent of computers, there has been a lot of research conducted in financial data, and specifically the use of machine learning (ML) techniques in financial time series data. However the general consensus has unfortunately been of a very pessimistic nature; citing the low signal to noise ratio[1] as being the primary source of failure in predictive power using ML techniques on financial time series data. One of the most influential advocates of this view has been the very well reputed name in quantitative finance, 'Seeking Alpha' (Marjanovic, 2018).

Our interest lies in assessing the professed challenges faced while attempting to use ML algorithms on financial data, and attempting to gauge whether it is indeed the flaws in the structure of the financial data that makes ML applications appear futile and powerless. We plan to ensure our research to be comprehensive in primarily two ways: first we consider the fact that a lot of the research has been made using relatively low-frequency data[2], and thus we plan to increase the granularity of the data and use higher frequency data[3] and compare the analysis of this with that of the generally used lower frequency data. Second we plan to run various ML algorithms, to rule out the possibility that a specific ML algorithm fails to capture the signals from the data, due to its individual shortcomings.

The principal algorithm we will run is a variant of Neural Networks, on 5-minute frequency, 5 years historical data of stock prices of 500 US stocks, providing a binary output. The choice of a classification algorithm is inspired by previous research conducted by machine learning practitioners and experts in the field of finance. Leung et al. (2000) provides substantial evidence that classification-based methods outperform level-based methods in the prediction of the direction of stock movement and trading returns maximization.

Our fascination with the outcome of this project is fairly obvious. ML techniques have developed an impressive reputation in various fields such as marketing, fraud detection and even self-driving cars. If our research is successful in displaying any positive signs towards predictive power in stock data, the successful algorithm could be potentially structured into a systematic trading strategy and implemented in financial markets.

1. Signal to Noise ratio will be defined and explained in greater depth in the Project Detail.

2. By referring to 'low frequency' data, we imply daily time series data.

3. The 'higher frequency' data refers to 5 minute time series data.

DETAILED PROJECT DESCRIPTION

Predicting future stock prices using past time series data has been one of the most challenging applications of machine learning. Stock prices prediction is regarded as a challenge since the stock market is essentially dynamic, nonlinear, complicated, nonparametric, and chaotic in nature (Abu-Mostafa & Atiya, 1996). Herein lays the need for a more complex learning algorithm: Neural Networks.

The application of artificial neural networks (ANNs) to time series methods have been explored in significant depth in the past (Faraway and Chatfield, 1998; Refenes, 1994; Trippi and DeSieno, 1992; Kaastra and Boyd, 1995) but their vulnerability to overfitting, convergence problems, and difficulty of implementation have proved to be persistent barriers. Thus the main algorithm presented in this paper is an adaptation of the ANNs; Deep Neural Networks (DNNs).

A DNN is an artificial neural network comprising of multiple hidden layers of units within the input and output layers, using a non-parametric approach to modeling based on minimizing an entropy loss function (Dixon and Klabjan, 2016). The precise implementation of the DNN algorithm in this project is explained now using 5 modules- the learning algorithm, cost function, parameters to be optimized, form of output, and finally the anticipated challenges for DNNs.

LEARNING ALGORITHM

Out project will make use of a feed-forward neural network (FFNN) to train the data and form the neural net architecture. The objective of a FFNN is to approximate some function f*. For instance, for a classifier, y = f*(x) maps an input x to a category y. A FFNN defines a mapping y = f(x;θ) and learns the value of the parameters θ that result in the best function approximation, through several layers. These models are termed feedforward since information flows through the function being evaluated from x, through the intermediate computations (layers) used to define f, and finally to the output y (Goodfellow-et-al, 2016).

In a fully connected feed-forward network, each node is connected to every node in the next layer as can be seen in the figure below.

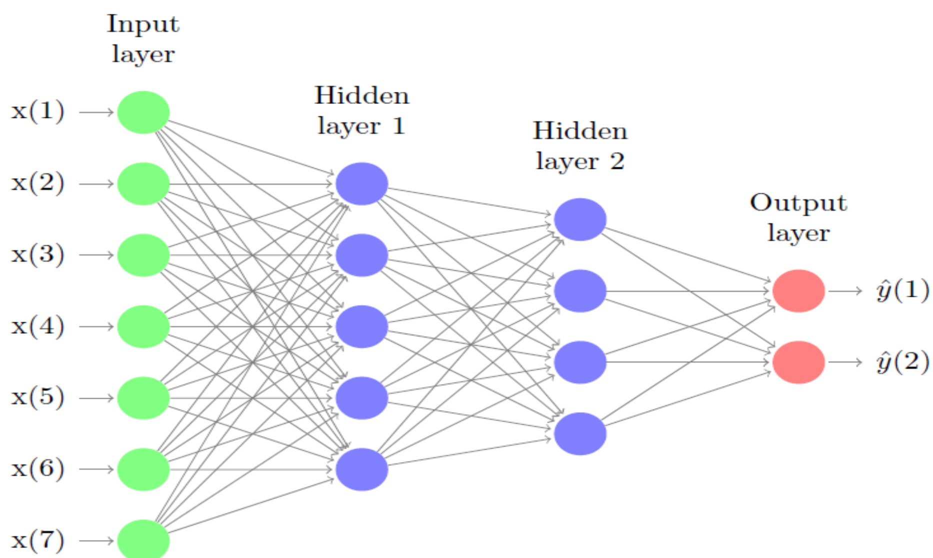

Figure 1: An illustrative example of a feed-forward neural network with two hidden layers, seven features and two output states. (Dixon and Klabjan, 2016)

The weights in each of the layers are updated with matrix-matrix products for each layer

$$\Delta \mathbf{w}^{(l)} = \gamma X^{(l-1)} \left(\delta^{(l)}\right)^{T}$$

For a fully connected feed-forward network, the output $\hat{y}$ is then the weighted sum of outputs from the previous layer l -1 that connect to node j in layer l:

$$\hat{y} = \sum_{i=1}^{n(l-1)} \omega_{ij}^{(l)} x_i^{(l-1)} + bias_j^{(l)}$$

COST FUNCTION

In order to find optimal weightings $\mathbf{w} := \{\mathbf{w}^{(l)}\}_{l:=1\to L}$ between nodes in a fully connected feed forward network with L layers, we seek to minimize a cross-entropy function of the form

$$E(\mathbf{w}) = -\sum_{n=1}^{N_{\text{test}}} e_n(\mathbf{w}), \qquad e_n(\mathbf{w}) := \sum_{k=1}^{K} y_{kn} ln\left(\hat{y}_{kn}\right).$$

In a more compact manner, the cross entropy function can be represented as:

$$C = -1n \sum x[yln(a) + (1-y)ln(1-a)]$$

Where *a* is the output, n is the total training data.

Two properties in particular primarily indicate the significance of cross entropy as a great cost function. First, it's non-negative. For instance, notice that: (a) all the individual terms in the sum in the above function are negative, since both logarithms are of numbers in the range 0 to 1; and (b) there is a minus sign out the front of the sum.

Second, if the neuron's actual output is close to the desired output for all training inputs, xx, then the cross-entropy will be close to zero. To see this, suppose for example that $y=0$ and $a\approx 0$ for some input x. This is a case when the neuron is doing a good job on that input. We see that the first term in the expression above for the cost vanishes, since $y=0$, while the second term is just $-\ln(1-a)\approx 0-\ln(1-a)\approx 0$. A similar analysis holds when $y=1$and $a\approx 1$. And so the contribution to the cost will be low provided the actual output is close to the desired output (A. Nielsen, 2015).

PARAMETERS

There are many training parameters we would be considering in our DNN, such as the size (number of layers and number of units per layer), the learning rate and initial weights. Conducting a grid search on these parameters to obtain optimality would be ideal, however very computationally expensive. Hence we aim to begin our analysis with a rather simple architecture with standard parameters such as 5 fully connected layers, ~20 neurons per layer, and a binary output. This applies to the implementation for each stock, and will be run on our database of 500 stocks. Going forward we will attempt to optimize some of these parameters, most significantly the learning rate, depending on the constraints given by time and computational power.

OPTIMIZATION ALGORITHM

Given the complexity of the neural net learning algorithm coupled with the huge database we plan to utilize, it was absolutely necessary to settle on a speedy optimization algorithm. Our proposed optimization method is to use a combination of mini-batching and stochastic gradient descent (SGD). SGD forms the backbone for the backpropagation learning algorithm, which is all too significant for the neural nets. SGD serves as the optimization algorithm of choice for DNNs due to the highly non-convex form of the utility function (Li et al. (2014). After random sampling of an observation i, the SGD algorithm updates the parameter vector w(l) for the lth layer using

w(l) = w(l) ⍰ rEi(w(l))   where ⍰ is the learning rate.

As for the mini-batching, past research has proven in a very convincing manner that mini-batching improves the computational performance of the feedforward and back-propagation computations (Shekhar and Amin, 1994). We process a specified set of observations in one mini-batch. This results in a change to the SGD algorithm and the dimensions of data-structures that are used to store variables. Mini batching crucially acts as the midway between gradient descent and pure SGD, merging the benefits of both worlds. It is a manner of methodology that is a compromise injecting enough noise to each gradient update, while achieving a relatively speedy convergence.

ANTICIPATED CHALLENGES

As mentioned earlier, the major challenged outlined by past research is that of the low signal to noise ratio in financial time series data. Low signal to noise ratio implies distorted datasets that are hard to predict simply for the reason that any forecastable pattern is camouflaged by randomness. This especially forms a challenge in financial time series data due to the variation in beliefs and expectations of market participants. This leads to the absence of any regularity in the data and gives rise to a lot of random trades in the market. In fact the whole branch of quantitative finance hypothesizes that all stock prices follow a random stochastic price process.

The other primary barrier we anticipate is that of computing power. We are extremely mindful of the huge computing power we require here, and have planned to fine-tune our algorithms in an attempt to contain this issue. Some other issues we face are centered around the data we have; the 5-minute frequency data[4] we plan to use is relatively of very good value. But one of the challenges in this dataset is that of the absence of adjusted close prices. Stock prices are known to fluctuate vigorously at certain times due to corporate announcements like dividends, stock splits, bonus issues etc.

These do not actually change the fundamental total value of the company in usual cases, but tend to alter prices rather radically. These could be the outliers in our analysis, and we would perhaps have to consider some outlier detection and removal method to deal with this issue.

[4] Data Sources: Bloomberg and Quandl.

COMPARISON WITH OTHER ALGORITHMS

Finally we plan to assess the efficacy of at least a couple of other algorithms such as Principle Component Analysis (PCA) and linear regression, depending on our time constraints. We would like to draw a comparison analysis in an attempt to discern which algorithm, if any, is most effective at absorbing and learning from the financial time series data. For certain algorithms, for instance PCA, we would require additional data such as the fundamental factors data, which we hope to retrieve from Bloomberg.

REFERENCES

-Marjanovic, B. (2018). *Don't Be Fooled By Stock Market Noise*. [online] Seeking Alpha. Available at: https://seekingalpha.com/article/3074086-dont-be-fooled-by-stock-market-noise [Accessed 13 Mar. 2018].

-M. Leung, H. Daouk, and A.Chen. Forecasting stock indices: A comparison of classification and level estimation models. International Journal of Forecasting, 16(2):173{190, 2000.

- Abu-Mostafa, Y. S., & Atiya, A. F. (1996). Introduction to financial forecasting. Applied Intelligence, 6(3), 205–213.

-L. Kaastra and M. S. Boyd. Forecasting futures trading volume using neural networks. Journal of Futures Markets, 15(8):953{970,1995. ISSN 1096-9934. doi: 10.1002/fut.3990150806. URL http://dx.doi.org/10.1002/fut.3990150806.

- A.P. Refenes. Neural Networks in the Capital Markets. John Wiley & Sons, Inc., New York, NY, USA, 1994. ISBN 0471943649.

-R. R. Trippi and D. DeSieno. Trading equity index futures with a neural network. The Journal of Portfolio Management, 19(1):27{33, 1992.

- Matthew Dixon, Diego Klabjan, and Jin Hoon Bang (2016). Classification-based Financial Markets Prediction using Deep Neural Networks.

-Goodfellow-et-al-2016, Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press,2016.

- Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

- M. Li, T. Zhang, Y. Chen, and A. J. Smola. Efficient mini-batch training for stochastic optimization. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, pages 661{670, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9}.

- V. K. S. Shekhar and M. B. Amin. A scalable parallel formulation of the backpropagation algorithm for hypercubes and related architectures. IEEE Transactions on Parallel and Distributed Systems, 5:1073{1090, 1994}.

Collaboration plan:

Task	Leader(s)	Deadline	Importance	Potential challenges
Research state of the art	All	03/25	Realized the relevance of the project	Identify successful applications of machine learning in Financial Markets Prediction
Read Deep Neural Networks Material	All	03/25	Good background for all members	Find and understand the theoretical concepts behind DNN
Select Algorithms to implements besides DNN	Eduardo, Rachit	03/31	As reference of the performance of Machine learning algorithms	A bad selection can lead to a bad representation of the potential of machine learning in Financial Markets Prediction
Get data and selected the sets for traning, test and validation	Rachit, Kshitij	4/07	Data is the pinnacle of machine learning	A good selection of the data set will impact directly to the accuracy of the outcome
Choose Libraries for implementation	Kshitij, Alberto	4/07	A good selection of the libraries can simply the implementation	The documentation of the selected libraries should be good enough to overcome any further challenges
Process data	Eduardo, Rachit		The different features and the rate of the data should be adapted to use in our implementation	Artificial events can present undesired deviations in the data
Implement DNN	Kshitij, Alberto	4/15	Base of the project	Select and implement the optimal cost functions
Implement other Machine learning algorithm	Eduardo, Alberto	4/15	As reference of the performance of Machine learning algorithms	
Get and analyze the results of the different algorithms	All	4/16	The analysis of the results of the different algorithms will lead us to the conclusions about the potential benefits of machine learning in Financial Markets Prediction	The selection of the criteria to analyze and compare the results is crucial for the presentation of our findings.
Write final results	Kshitij, Eduardo	4/18	Put all the findings in words for the final	To be able to express in the best way the results, the

			report	analysis and the criteria for the comparison
Poster design	Alberto, Rachit	4/21	A visual representation of the project	To find an balance design to show our findings
Print Poster	Eduardo	4/22		To have time enough to solve any inconvenient in order to have the best quality
Send final report	Alberto	4/24	The style correction and proofreading is essential for a successful report	To have time enough and be sure everything make sense