# A critique of the standard neural network application to financial time series analysis

Michael de la Maza and Deniz Yuret
Numinous Noetics Group
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Room NE43-815
545 Technology Square
Cambridge, MA 02139
mdlm@ai.mit.edu, deniz@ai.mit.edu

**Abstract**

Neural networks are one of the most widely used artificial intelligence methods for financial time series analysis. In this paper we describe the standard application of neural networks and suggest that it has two shortcomings. First, backpropagation search takes place in sum of squared errors space instead of risk-adjusted return space. Second, the standard neural network has difficulty ignoring noise and focusing in on discoverable regularities. Both problems are illustrated with simple examples. We suggest ways of overcoming these problems.

## 1 Introduction

Ever since McCulloch and Pitts [8] published their landmark paper describing a neural calculus, researchers have explored how biologically inspired networks might be employed to solve a wide class of problems. In recent years, Wall Street, in its never ending search for new ways to beat the market, has turned to neural networks as a possible answer. The attention focused on neural networks for financial time series analysis stems from:

- An enormous interest in the academic community which springs, in part, from the misguided belief that neural networks are similar to the human brain in important ways. Neural networks have been shown to have some very desirable theoretical properties (see, e.g., [2]) which have made them the tool of choice for many inference tasks.
- The non-parametric nature of neural networks. Researchers who want to "let the data speak for itself" prefer neural networks to standard statistical methods. The fact that "non-parametric" is sometimes read as "no search bias" is an important component of this appeal.
- The accessibility of neural networks. There are dozens of companies and hundreds of consultants who purport to understand how to beat the markets with turn-key neural networks.

Despite this interest there are few known instances of success with neural networks in real markets. LBS Capital Management, which is on the Inc. 500 list of America's fastest growing companies, manages part of its portfolio using neural networks. The performance of at least two Fidelity funds run by Bradford Lewis, the Disciplined Equity Fund and the Small-Cap Fund, is due in part to neural networks. However, these neural networks are probably not fully autonomous and the final trading decisions are probably made after considerable human intervention.

One reason for this lack of apparent success might simply be that companies which have met with success are reluctant to publicize it for fear of attracting competitors. However, the academic community does not labor under any such restriction and, in the past, when new approaches were applied with success to financial time series analysis, these results were published quickly. In this paper we suggest that there might be another explanation for these skimpy results: neural networks, at least as they are typically applied, are not well-suited for financial time series analysis. Section 2 of this paper describes the standard neural network application. Section 3 discusses the two shortcomings of the standard application that we focus on in this paper. Section 4 outlines proposed, but not implemented, solutions to these problems, and section 5 summarizes the conclusions of the paper.

# 2 The standard neural network application

Before discussing the two shortcomings of neural networks, we must first outline the standard neural network application. This "standard" was created by averaging several dozen applications of neural networks that appear in the applied computational finance literature (see, e.g., [1]). More complete and general discussions of neural networks can be found elsewhere (e.g., [9, 10]).

The standard neural network consists of three sequential layers of nodes in the following order: input layer, hidden layer, and output layer. Via connections, each node in the hidden layer receives an input from each node in the input layer and each node in the output layer receives an input from each node in the hidden layer. To produce the output, a weighted sum of these inputs is passed through a transfer function. In the typical application, the transfer function is the sigmoid $(1+\exp^{-s})^{-1}$ where $s$ is the sum of the inputs into a node. Each connection between the input layer and the hidden layer and the hidden layer and the output layer has a weight which is updated by the backpropagation procedure which performs a gradient descent search in sum of squared errors space.

In the typical neural network application, technical and fundamental data are given as input into the neural network and the price or the change in price of some financial instrument is predicted at some point in the future. Sometimes the volatility or change in volatility is also predicted.

# 3 Two problems with the standard application

This section describes two shortcomings of the standard neural network application to financial time series analysis. First, we note that stock market participants are not primarily interested in predicting stock market prices, but rather they are interested in maximizing some measure of risk-adjusted return, such as the Sharpe ratio. We discuss the distinction and note that the standard neural network application is restricted to searching in sum of squared errors space and extending it does not appear to be straightforward. We give an example of a strategy which minimizes the sum of squared errors but has a sub-optimal risk-adjusted return, as measured by the Sharpe ratio.

Second, we note that neural networks are forced to make a prediction at every point of a time series. As a result, they are incapable of explicitly ignoring sections of the data that do not have discoverable regularities. This causes them to employ their representational capacity in regions of the space where it is unnecessary and this reduces their performance on the component of the data that does have discoverable regularity. We illustrate this shortcoming with a simple artificial time series.

## 3.1 Sum of squared errors vs. risk-adjusted return

> My financial success stands in stark contrast with my ability to forecast events.
>
> - George Soros[11, page 301]

We present two trading strategies which have the property that one is superior when sum of squared errors is the utility measure but is inferior when risk-adjusted return, as defined by the Sharpe ratio, is the utility measure. Because what matters to investment managers and their clients is risk-adjusted return, this suggests that the standard neural network application, which minimizes the sum of squared errors, should be retooled.

Consider the time series: 10,20,30,60,10,20,30,10. This can be thought of as the price of a security on consecutive days. If the neural network is asked to predict the next element in this series given the three previous elements then there are a total of four unique input sequences and five input/output pairs, as shown

| Sequence (input) | Next element (output) | Strategy A Prediction | Strategy A Profit | Strategy B Prediction | Strategy B Profit |
|---|---|---|---|---|---|
| 10,20,30 | 60 | 35 | 30 | 30 | 0 |
| 20,30,60 | 10 | 10 | 50 | 10 | 50 |
| 30,60,10 | 20 | 20 | 10 | 20 | 10 |
| 60,10,20 | 30 | 30 | 10 | 30 | 10 |
| 10,20,30 | 10 | 35 | -20 | 30 | 0 |

Table 1: The five sequences in the time series 10,20,30,60,10,20,30,10. The neural network is asked to predict the next element in the sequence given the previous three elements. Strategy A is the prediction made by the strategy which minimizes the sum of squared errors. Strategy B is a strategy which has a higher sum of squared errors, but a lower Sharpe ratio, than strategy A, as shown in Table 2.

| | sum of squared errors | $\mu$ | $\sigma$ | Sharpe ratio ($\frac{\mu}{\sigma}$) |
|---|---|---|---|---|
| Strategy A | 1250 | 16 | 23.324 | 0.686 |
| Strategy B | 1300 | 14 | 18.547 | 0.755 |

Table 2: Comparison of strategy A and strategy B. Strategy A has a lower sum of squared errors, but strategy B has a better Sharpe ratio (the risk-free rate is assumed to be 0%).

in Table 1. This table shows two strategies, strategy A and strategy B, which produce the same output for three of the four input sequences, but differ on the only sequence which is not followed by a unique element (the first and last rows in Table 1). Strategy A is the optimal sum of squared errors strategy but strategy B has a better Sharpe ratio, as shown in Table 2. This comparison assumes that the strategy buys when the predicted value of the next element is greater than the last element of the sequence, sells when the predicted value of the next element is less than the last element of the sequence, and does not trade when these two values are the same. A constant amount is traded each time and the market is assumed to be frictionless.

A skeptic would argue that the flaw in this example is that a *forecasting* strategy should not be confused with a *trading* strategy. Such a skeptic would say that once the neural network has been trained to be a good *forecasting* strategy then some sort of post-processor should be added to the output to turn it into a good *trading* strategy. This appears to be a somewhat circuitous approach: Why search in the wrong space (sum of squared errors) and then repair, when the search can be done in the space of interest (risk-adjusted return)?

## 3.2 The importance of knowing what you don't know

The standard application of neural networks forces a prediction to be made for every input sequence. Because of this, the neural network must distribute its representational capacity across the entire time series, instead of being able to focus in on regions which have discoverable regularity. As a result, the neural network is sometimes incapable of uncovering simple regularities because its representational capacity is inappropriately employed.

This point is illustrated by the function shown in Figure 1(A) which has three segments: the first (points 0 to 49) and third (points 100 to 149) segments were generated by randomly choosing numbers between 0.25 and 0.75 and the second segment (points 50 to 99) is the line $y = x/100 - .25$. A wide variety of neural networks with varying numbers of hidden units (0, 2, 5, 10, 20, and 50), different learning rates (.01, .001, and .0001), and different inertia parameters (0, .1, .2, .3, .4, .5, .6, .7, .8, and .9) were trained to predict the value of point $n$ given points $n-5$ through $n-1$. The predictions made by one representative neural network are shown in Figure 1(B). The predictions made by the same neural network when trained only on the second segment of the function shown in Figure 1(A) are shown in Figure 1(C).

A comparison of these two graphs shows that the prediction of the middle segment is much more accurate in Figure 1(C) than in Figure 1(B). Why? Because part of the neural network's representational capacity has

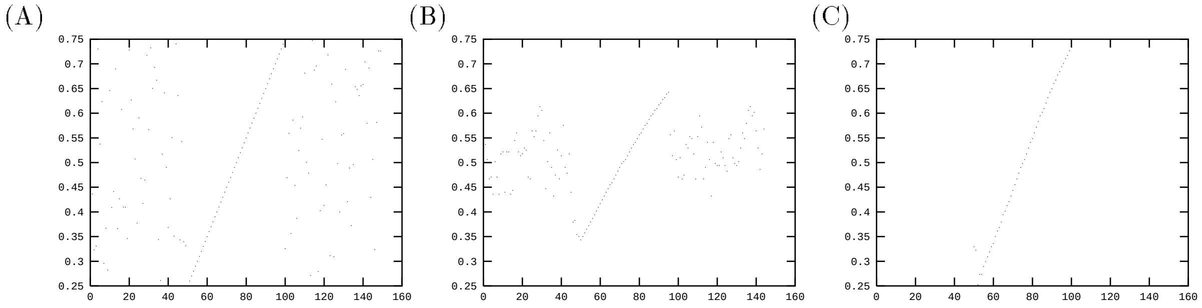


Figure 1: (A) Target function. The neural network is asked to predict the value of point $n$ and is given as input the value of points $n-5$ through $n-1$. The first 50 points are random numbers between 0.25 and 0.75 as are the last 50 points. The middle 50 points satisfy the linear equation $y = x/100 - .25$. (B) Output of a neural network with five hidden units trained on the target function. The inertia parameter was set to 0.9 and the learning rate was set to .001. The neural network was trained for one million epochs. This output graph is representative of the performance of many neural networks that we tested on this function. (C) Output of a neural network trained with five hidden trained on the middle 50 points of the target function. The inertia parameter was set to 0.9 and the learning rate was set to .001. The network was trained for one million epochs. Compare to the middle 50 points in (B).

been spent trying to model the first and third segments, even though these segments do not have discoverable regularity. Importantly, the function in Figure 1(A) is extremely simple. The regularities in real financial time series are significantly more subtle and the ability to distinguish between the knowable and unknowable becomes significantly more important.

The underlying problem on which this example stands is the requirement that the standard neural network make a prediction at every data point: "don't know" is not an option. However, not only does the capability to say "don't know" free up representational capacity, but, as in the case of Figure 1(A) over the interval [0,49] and [100,149], it is sometimes the best possible model of the data given the representational capacity of the neural network. Many statistical methods, such as linear and multiple regression, share this shortcoming with the standard neural network application. As a result, these methods are, in effect, unable to say what human traders often do: "Current market conditions are beyond my comprehension, therefore I am not going to try to understand them." For the graph shown in Figure 1(A) this is *the* appropriate response for the first and third segments.

# 4 Possible solutions

Our view is that the shortcomings discussed in the previous section are potential showstoppers which demand the full attention of those who are interested in applications of neural networks to financial time series analysis. In this section we speculate on ways of overcoming these limitations.

One way to address both problems is to change the search algorithm from backpropagation to simulated annealing [7, 5], genetic algorithms [4, 3], standard gradient descent, or any one of the many optimization methods which allow search over arbitrary utility functions. Adopting this solution would allow the direct implementation of risk-adjusted return as the utility function and the neural network would no longer be forced to make a prediction for every input. Under this scheme, the representation (or model) would still be sums of nested sigmoidal functions.

Finessing the data so that minimizing the sum of squared errors is equivalent to maximizing risk-adjusted return is another way to address the first shortcoming. Success in finding such a transformation would probably be helpful in finding solutions to other problems not associated with financial time series analysis.

A possible solution to the second problem has been explored by Jordan and Jacobs [6]. They suggest creating a hierarchical mixture of neural networks so that each neural network can become an expert at identifying the regularities in one component of the data.

## 5 Conclusions

The main results of this paper are:

- *Neural networks are better suited for forecasting (minimizing the sum of squared errors) than for trading (maximizing risk-adjusted return).*
- *The inability to generate "don't care" predictions hinders the ability of neural networks to focus in on discoverable regularities.*

These results suggest that the standard application of neural networks which permeates the financial time series analysis literature may be flawed. Because the standard neural network does not search over the space that market participants care about, they produce strategies that can only be optimal by chance. Given that making money in the market is so difficult, learning algorithms should at the very least be programmed to search over the correct space. In addition, because neural networks are incapable of ignoring sections of time series data that do not have discoverable regularity, they are sometimes incapable of exposing very simple regularities in artificial data, as illustrated by one, admittedly concocted, example. Our personal view is that the these shortcomings are grave enough to force a re-evaluation of the standard neural network application to financial time series analysis.

## 6 Acknowledgments

Thanks to Steve Baron, Tom Berghage, Jim Hutchinson, and Michael Jones. Some of the neural network simulations were performed on the Cray C90 at the Pittsburgh Supercomputing Center. This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-92-J-4097 and by the National Science Foundation under grant number MIP-9001651.

## References

[1] E. M. Azoff. *Nueral Network Time Series Forecasting of Financial Markets.* John Wiley & Sons, Inc., New York, 1994.

[2] A. Barron. Universal approximation bounds for superpositions of a sigmoidal function. Technical Report 58, University of Illinois at Urbana-Champaign, Champaign, IL, 1991.

[3] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[4] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[5] L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and Computer Modeling*, 18(11):29–57, 1993.

[6] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. Technical Report 1440, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1993.

[7] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[8] W. S. McCulloch and W. H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[9] S. K. Rogers and M. Kabrisky. *An Introduction to Biological and Artificial Neural Networks for Pattern Recognition.* SPIE Optical Engineering Press, Bellingham, WA, 1991.

[10] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, Cambridge, MA, 1986.

[11] G. Soros. *The Alchemy of Finance: Reading the Mind of the Market.* John Wiley & Sons, Inc., New York, 1994.