Neural Network and Genetic Algorithm Based Trading Systems

Donn S. Fishbein, MD, PhD
Neuroquant.com

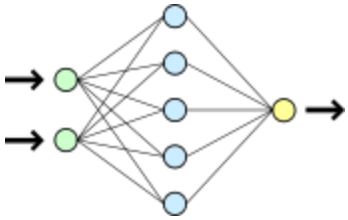
Consider the challenge of constructing a financial market trading system using commonly available technical indicators. There are thousands of indicators available, most of which have one or more fixed parameters. Having selected appropriate indicators and parameters, rules must be constructed that translate an indicator into a buy, sell or hold decision. If more than one indicator is used, more than one rule may result, and another method for combining the rules into a single transactional decision is needed.

One way to look at this problem is to consider that one is mapping a series of inputs (indicators and their parameters) to an output (buy, hold or sell). The problem could be thought of as finding the coefficients for a multi-variable polynomial equation in the form:

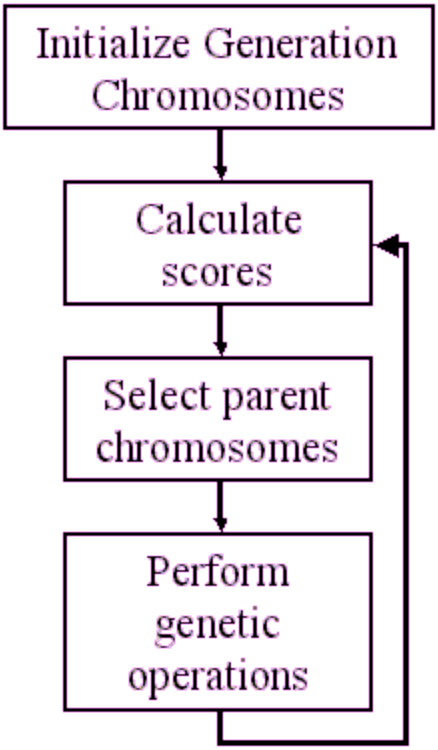
$$f(x,y,\ldots) = a_n x^n + a_{n-1} x^{n-1} + \ldots a_1 x + a_0 + b^n y_n + \ldots$$

where x, y and so forth represent inputs such as price or technical indicators. There are a number of ways to fit the coefficients of the above equation to the data. Brute force methods suffer from an exponential rise in the time required as the number of coefficients increases. While a number of methods to approximate coefficients are available, the remainder of this paper will discuss two approximation methods which are derived at least in part from biological models: genetic algorithms and artificial neural networks.

Artificial neural networks are modeled after the neuronal architecture of the human brain. Interconnected processing nodes are organized in two or more layers and work in parallel to process input data. Each intranodal connection is weighted, and by adjustment of these weights a network is able to learn and store information. Numerous neural network architectures exist. The feed-forward back-propagation model has been well characterized, and is commonly employed in financial market prediction. In the schematic below, input nodes are shown in green, hidden nodes in blue, and an output node in yellow. Information flows from left to right in the network pictured.

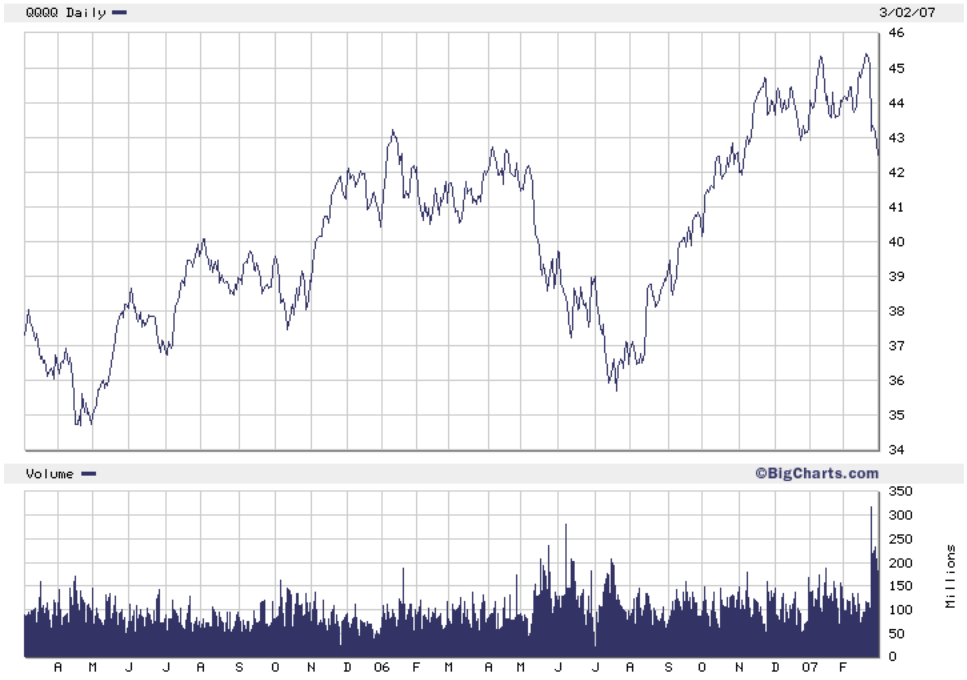


Genetic algorithms can be used to find approximate solutions for difficult to solve problems. Fashioned using concepts from biological evolution, such as inheritance, crossover, mutation, and extinction, genetic algorithms can examine a broad field of candidates and evaluate potential solutions for their fitness to solve the problem at hand. The algorithm may converge on an acceptable solution in less time than a brute force search.

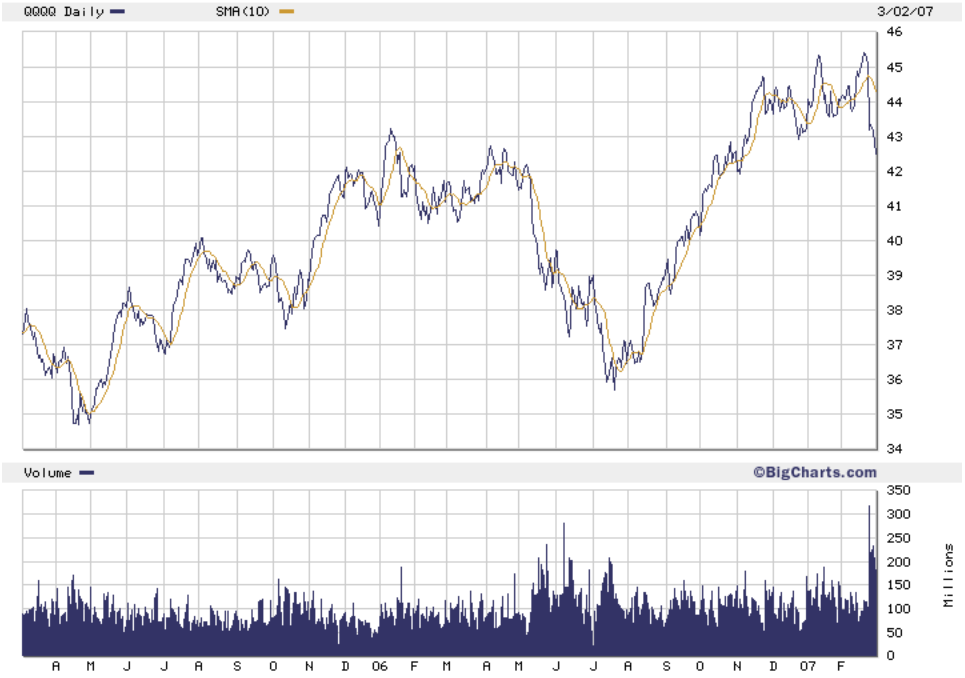


Although there is overlap between the capabilities of artificial neural networks and genetic algorithms, their combined capabilities can be synergistic. A common approach in financial market prediction is to use a genetic algorithm to select inputs and parameters for inputs, and a neural network to transform these inputs into a trading decision e.g., long, cash, sell. The process looks similar to the GA flowchart above, except that each “calculate scores” step involves the training of a neural network with the then current inputs as selected by the genetic algorithm.

Assume a technically competent but financially naive individual wishes to devise a mechanical (objective) system for trading a financial market. For the sake of argument, we'll use the Nasdaq-100 tracking exchange traded fund QQQQ, and initially examine the two year period ending in March, 2007.



At first glance, the problem doesn't seem overwhelming. A single time series is present, and the major task is to predict its future direction. The price chart looks quite noisy. Although clear trends are evident when looking at this long term chart retrospectively, the volatility of the curve makes day to day trading decisions difficult. One observation, perhaps supported by Fourier analysis, is that the noise has a significant high frequency component. Perhaps filtering some of the high frequency noise would be helpful. A simple moving average, the average of the n-preceding prices, acts as a low pass filter and reduces high frequency noise. The graph below adds a 10-day simple moving average (yellow line).



At first glance, this seems to be an improvement. The SMA(10) line is smoother and shows more easily recognizable trends. Irrelevant short term changes in the direction of the original price signal are eliminated. Since the simple moving average line lags the price line, perhaps one trading strategy would be to trade long when the price rises above the simple moving average, and short for the reverse condition, expressed as:

```
Long: Price > SMA(10);  Short: Price < SMA(10)
```

However logical this seems, the results are disappointing:

	Annual Return	%Profitable	%Drawdown
Fixed Parameters & Rules			
Price > SMA	-10.7%	32%	-47.1%

The system loses money, and at its worst point suffers a decline of nearly 50%. There are several possible reasons for this performance.

Perhaps the parameter chosen for the simple moving average is not optimal, or perhaps a single relationship cannot adequately characterize the market. What if a second longer term simple moving average is introduced? In general, in a smoothly rising market the price will first cross above the shorter term simple moving average, and then the shorter term simple moving average will cross above the longer term moving average, suggesting the following trading rules:

```
Long: Price > SMA(10) AND SMA(10) > SMA(50)
Short: Price < SMA(10) AND SMA(10) < SMA(50)
```

While this seems a logical improvement, the results are improved but still disappointing:

	Annual Return	%Profitable	%Drawdown
Fixed Parameters & Rules			
Price > SMA	-10.7%	32%	-47.1%
Price > SMA, SMA_1 > SMA_2	5.4%	41%	-28.5%

Continuing with this progression, a third longer term simple moving average is added. Note that the three parameters chosen, 10, 50 and 200 days, represent commonly used parameters for short term, intermediate term, and long term periods. The rules become:

```
Long:  Price > SMA(10) AND SMA(10) > SMA(50) AND SMA(50) > SMA(200)
Short: Price < SMA(10) AND SMA(10) < SMA(50) AND SMA(50) < SMA(200)
```

and the results become:

	Annual Return	%Profitable	%Drawdown
Fixed Parameters & Rules			
Price > SMA	-10.7%	32%	-47.1%
Price > SMA, SMA_1 > SMA_2	5.4%	41%	-28.5%
Price > SMA, SMA_1 > SMA_2 * 2	-13.9%	14%	-38.6%

Apparently, the answer is simply not “more rules.” Either simple moving averages are not helpful in predicting future prices, or the parameters for the simple moving averages were selected poorly, or the rules were chosen poorly. If there is some predictive value in simple moving averages, perhaps genetic algorithms and/or neural networks could help develop better trading rules using simple moving averages.

Consider first that perhaps either the parameters selected or the

number of rules employed are suboptimal. A genetic algorithm could be used to select to optimize these choices. No alteration is made to the structure of the buy and sell rules; the genetic algorithm simply selects the best parameters for each simple moving average and determines whether inclusion of each rule is beneficial. Each system is optimized over an 42 month period and then tested over a six month period. With this optimization performed, the following results are seen:

	Annual Return	%Profitable	%Drawdown
Fixed Parameters & Rules			
Price > SMA	-10.7%	32%	-47.1%
Price > SMA, SMA_1 > SMA_2	5.4%	41%	-28.5%
Price > SMA, SMA_1 > SMA_2 * 2	-13.9%	14%	-38.6%
GA Only			
Price > SMA	1.1%	47%	-9.3%
Price > SMA, SMA_1 > SMA_2	26.5%	54%	-6.4%
Price > SMA, SMA_1 > SMA_2 * 2	6.8%	56%	-3.5%

Improvement is noted in all performance categories. The outsized return when two rules are employed might represent a statistical anomaly, and further testing using financial instruments other than QQQQ and time periods other than 2003 to 2007 would be necessary to conclude the improvement was real.

Turning to neural networks, one is trained for each set of inputs, using the fixed parameters for each simple moving average and employing the closing price and one, two or three simple moving averages as before. This time, no fixed rules are used. Rather, the neural network is given the inputs and constructs a mapping between the inputs and outputs. Once again 42 months of data is used to train the network and the remainder of the data employed as an out-of-sample test. The results generated are:

	Annual Return	%Profitable	%Drawdown
Fixed Parameters & Rules			
Price > SMA	-10.7%	32%	-47.1%
Price > SMA, SMA_1 > SMA_2	5.4%	41%	-28.5%
Price > SMA, SMA_1 > SMA_2 * 2	-13.9%	14%	-38.6%
GA Only			
Price > SMA	1.1%	47%	-9.3%
Price > SMA, SMA_1 > SMA_2	26.5%	54%	-6.4%
Price > SMA, SMA_1 > SMA_2 * 2	6.8%	56%	-3.5%
NN Only			
Price	-15.8%	0%	-14.0%
Price SMA	-12.6%	33%	-14.0%
Price SMA x 2	-15.8%	0%	-14.0%
Price SMA x 3	0.0%	50%	-10.3%

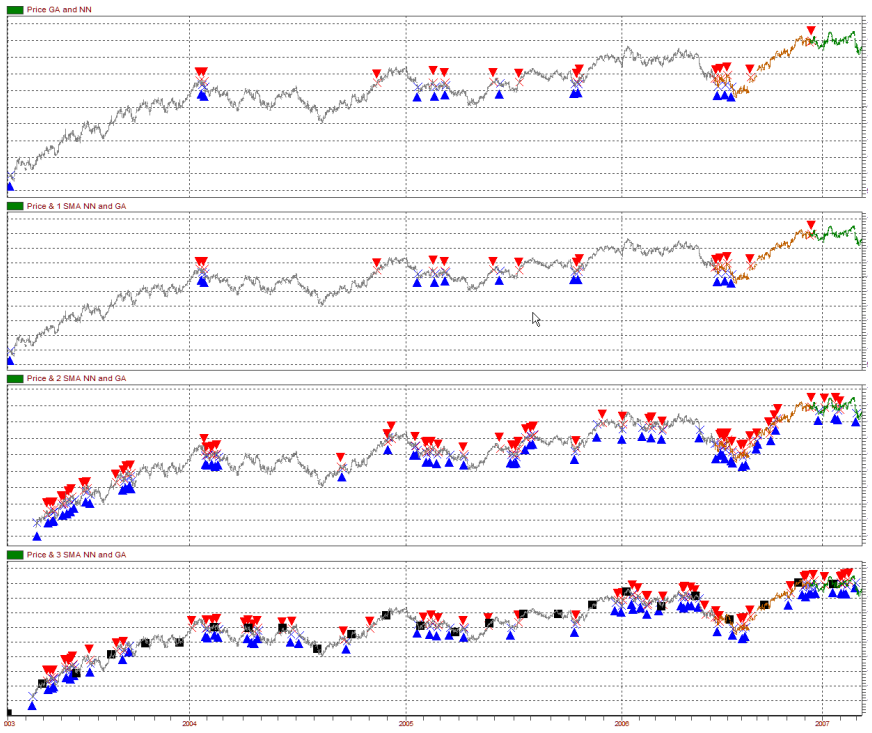
In this case, using only a neural network with a fixed number of inputs with fixed parameters did not improve on the first iteration using fixed parameters and rules. Perhaps not enough data was present to allow the neural network to deduce the same fixed rules used in the first example, or maybe the fixed parameters did not fit the characteristics of the input data.

Finally, neural networks and genetic algorithms are used together. A genetic algorithm selects the best parameters for each input, and the best inputs (among the candidate inputs chosen) for each network. The optimization procedure is summarized as:

```
Choose initial sets of inputs
Evaluate the fitness of each set of inputs in the population
    Repeat
        Train neural network with each set of inputs
        Select best-ranking set of inputs to reproduce
        Breed new generation through crossover and mutation
        Evaluate the individual fitnesses of the offspring
        Replace worst ranked part of population with offspring
    Until desired fitness is obtained
```

The underlined line in the pseudocode above represents the neural network contribution, while the remainder shows the genetic algorithm. In the combination system, a series of candidate inputs is selected, along with their parameters. In this case the candidate inputs are the closing price and simple moving averages of 10, 50 and 200 day length. The system is free to select the optimal parameters, number of inputs, and

mapping between the selected inputs and the output decision.



The chart above shows the four combined neural network/genetic algorithm systems generated using price and price with one, two or three simple moving averages. Buy signals are shown by a blue up arrow, and sell signals by a red down arrow. The results generated by these systems are shown in the spreadsheet below:

	Annual Return	%Profitable	%Drawdown
Fixed Parameters & Rules			
Price > SMA	-10.7%	32%	-47.1%
Price > SMA, $SMA_1 > SMA_2$	5.4%	41%	-28.5%
Price > SMA, $SMA_1 > SMA_2 * 2$	-13.9%	14%	-38.6%
GA Only			
Price > SMA	1.1%	47%	-9.3%
Price > SMA, $SMA_1 > SMA_2$	26.5%	54%	-6.4%
Price > SMA, $SMA_1 > SMA_2 * 2$	6.8%	56%	-3.5%
NN Only			
Price	-15.8%	0%	-14.0%
Price SMA	-12.6%	33%	-14.0%
Price SMA x 2	-15.8%	0%	-14.0%
Price SMA x 3	0.0%	50%	-10.3%
GA and NN			
Price	11.0%	100%	-3.2%
Price SMA	11.0%	100%	-3.2%
Price SMA x 2	28.7%	88%	-4.5%
Price SMA x 3	23.5%	77%	-5.4%

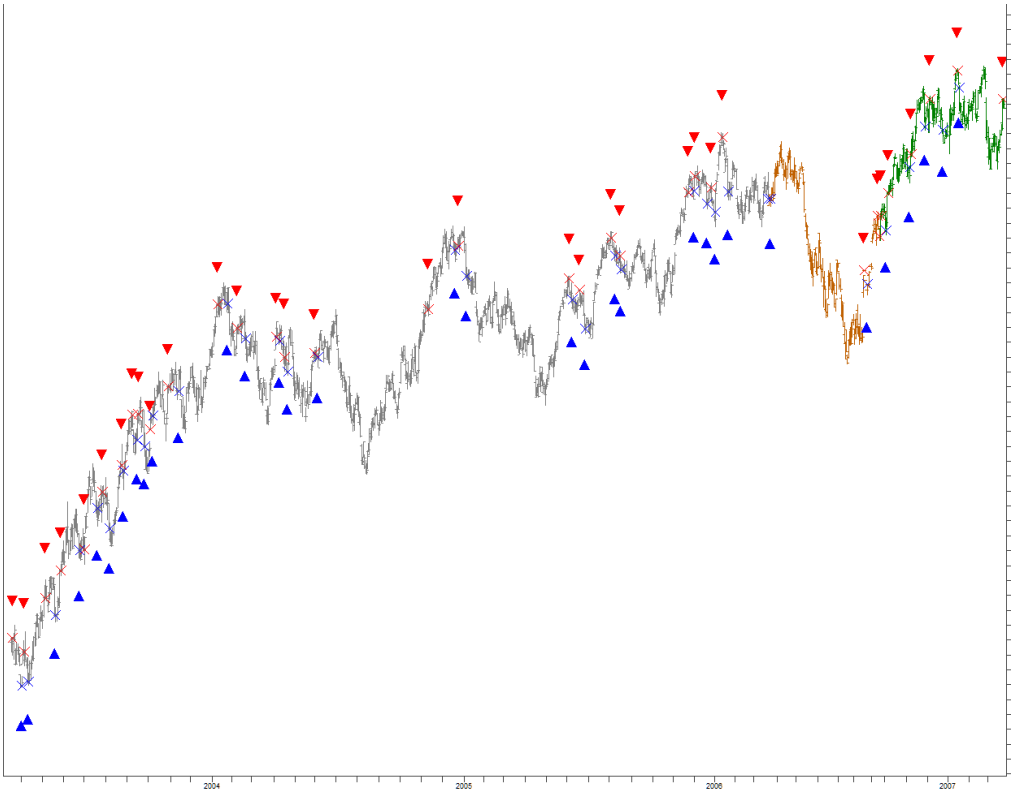
Significant improvement in performance is noted for most input sets. The results for price and price with one simple moving average are identical because the genetic algorithm did not find the addition a single simple moving average helpful, and therefore eliminated the single simple moving average as an input. A similar optimization did not occur when a third simple moving average was added to the input candidates, and the failure of the genetic algorithm to eliminate an unhelpful or redundant input degraded the ultimate neural network result.

Within the confines of this limited example, it is possible given sufficient time, that by trial and error a trading system could be developed as profitable as the the system shown above. On the other hand, the use of genetic algorithms and neural networks offer a systematic and automated system for the development of trading systems. The advantage becomes more apparent when more complicated systems are contemplated.

The final system which will be shown was developed using the following inputs:

- simple moving average
- exponential moving average
- adaptive (variable length) moving average
- MACD
- fast and slow stochastics (oscillators)
- linear regression line(s)
- momentum and acceleration indicators

In all, 18 separate parameters for 12 indicators were evaluated by a hybrid genetic algorithm/neural network system. Trading the Nasdaq-100 exchange traded fund QQQQ, the system was optimized for a 42 month period, and tested for a 6 month period on out-of-sample data ending in March, 2007. $0.01/share was charged for commissions. During the out-of-sample test period, the system earned an annualized return of 30.1%, with 74% of trades profitable and a maximum open trade drawdown of 18.1%. The trades generated during the optimization and testing periods are shown in the graph below.



In this case, the complexity of the input set would have made hand optimization very difficult. The hybrid genetic algorithm/neural network system required 13 minutes to fully train the system, with a high degree of confidence that with the given inputs the trading system generated was close to an optimal solution.

A number of issues regarding trading system design have not been discussed yet should be considered prior to live trading, including scientific validation, money management, and data preprocessing. With these issues factored in, hybrid genetic algorithm/neural network trading systems offer a unique methodology for the development of profitable trading systems for financial markets.

References

Arnold, Curtis M. Timing The Market. Weiss Research, Inc., 1993

Aronson, D. Evidence-Based Technical Analysis. John Wiley and Sons, 2007.

Barnes, RM. Trading System Analysis. McGraw-Hill, 1997.

Chen CH. Fuzzy Logic and Neural Network Handbook. IEEE Press, 1996.

Colby, RW. The Encyclopedia of Technical Market Indicators. McGraw-Hill, 2003.

Fishbein DS. Trading Financial Markets with Neural Networks and Genetic Algorithms. Trenton Computer Festival Proceedings, 2001.

Fishbein DS. Neural Networks and Genetic Algorithms: Another Tools for the Technical Analysis of Financial Markets. Trenton Computer Festival Proceedings, 2002.

Fishbein DS. Mechanical Stock Trading Systems That Really Work. Trenton Computer Festival Proceedings, 2003.

Fishbein DS. How to time the stock market Using Artificial Neural Networks and Genetic Algorithms. Trenton Computer Festival Proceedings, 2004.

Fosback, Norman G. Stock Market Logic. Dearborn Financial Publishing, Inc., 1991.

Kaufman PJ. New Trading Systems and Methods (4th Edition). John Wiley and Sons, 2005.

Murphy John. Intermarket Analysis. Wiley Trading, 2004.

Neuroquant.com: www.neuroquant.com

Zirilli, Joseph S. Financial Prediction Using Neural Networks. International Thomson Computer Press, 1997.

Zweig, M. Winning of Wall Street. 1986, 1997(revised), Warner Books.